UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

# DIARIO DE ENTRENADORES: APLICACIÓN PARA GESTIÓN DE EQUIPOS DEPORTIVOS

# COACH JOURNAL: APPLICATION FOR SPORT TEAMS MANAGEMENT


Realizado por
**Jairo Campaña Cañete**

Tutorizado por
**José Manuel Jerez Aragonés**
**Luis Manuel Llopis Torres**

Departamento
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2019

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

# Resumen

Tanto el entrenamiento como la gestión de un equipo deportivo pueden ser tareas complicadas, sobre todo, si se trata de un equipo infantil. A veces, puede no prestarse atención, bien por falta de tiempo o bien por falta de conocimiento, a las necesidades individuales de cada miembro del equipo.

Este proyecto tiene como objetivo mitigar este tipo de problemas, proporcionando a los entrenadores de equipos deportivos una herramienta cómoda, sencilla y accesible en forma de aplicación web multiplataforma, que les ayude a confeccionar entrenamientos personalizados y que les asista durante todo el proceso, desde los partidos hasta el momento de la gestión del grupo.

Con Diario de Entrenadores un usuario podrá crear un club deportivo, definir equipos dentro de este e invitar a otros usuarios a colaborar como entrenadores. Por cada equipo, se podrán añadir jugadores así como eventos que pueden ocurrir durante un partido y que se consideren importantes para su estudio y posterior análisis que tiene como objetivo la confección de un plan de entrenamiento.

Tras esto, el usuario podrá simular partidos en tiempo real y podrá anotar, por cada unidad de tiempo, los eventos que ocurren y el jugador que los ejecuta, en vista de identificar las virtudes y carencias de cada jugador del equipo. Por último, los resultados de los partidos pasarán a ser procesados por determinados algoritmos de análisis, que generarán una serie de gráficas para que el usuario los analice y pueda planificar los entrenamientos en base a ellos.

**Palabras clave:**

Deporte, Equipos, Entrenadores, Estadística, Aplicación Web, R, ASP.NET

# Abstract

Both training and management of a sports team can be complicated tasks, especially if it is a children's team. Sometimes, either due to the lack of time or the lack of knowledge, attention may not be paid to the individual needs of each team member.

The aim of this project is to mitigate this kind of problem by providing sports team coaches with a comfortable, simple, and accessible tool in the form of a multi-platform web application, which will help them to create personalized trainings and assist them throughout the whole process, from the matches to the group management.

With Coach Journal a user can create a sports club, define teams within it and invite other users to collaborate as coaches. For each team, players can be added as well as events that may occur during a match and that are considered of importance for their study and subsequent analysis for the preparation of a training plan.

After all this, the user can simulate matches in real time and can annotate, for each unit of time, the events that occur and the player who executes them, in order to identify the virtues and shortcomings of each player on the team. Finally, the results of the matches will be processed by certain analysis algorithms, which will generate a series of graphs for the user to analyze and plan the training based on them.

**Keywords:**

Sports, Teams, Coaches, Statistics, Web Application, R, ASP.NET

# Índice

# 1

# Introducción

Se comenzará la memoria y, en especial, este capítulo exponiendo las razones que motivaron este proyecto, así como los objetivos finales del mismo.

Tras esto, se detallarán las tecnologías utilizadas y, para concluir la introducción, se resumirá la estructura de la memoria.

## 1.1 Motivación

La idea del proyecto nace de uno de los tutores del mismo y de la falta de herramientas similares con la misma finalidad y flexibilidad.

Como se ha expuesto anteriormente en el resumen, gestionar un equipo deportivo, especialmente si es un equipo infantil, puede ser muy complicado. Conocer cuales son las virtudes, defectos y necesidades particulares de cada jugador y planificar un entrenamiento personalizado y adecuado no es una tarea sencilla.

Si se realiza una búsqueda en el mercado actual de aplicaciones para entrenadores, puede comprobarse a simple vista que no están pensadas para ayudar en este tipo de cometido.

Para empezar, la mayoría de aplicaciones se centran en el aspecto táctico del juego, ofreciendo herramientas como pizarras tácticas y ayuda para las jugadas a desarrollar durante un partido.

También podemos ver que dichas aplicaciones están pensadas para un deporte concreto y no se puede abstraer ni generalizar para más deportes, lo cual confiere el problema de que si se encuentra una aplicación con las características deseadas, puede no ser para el deporte concreto que se practica.

Y por último, si existe alguna aplicación semejante, se puede constatar que no ofrece los resultados requeridos y que carece de cierta flexibilidad y usabilidad para el usuario.

Debido a todo esto, surge la necesidad y la idea de disponer de una herramienta cómoda, sencilla y accesible que sirva como ayuda a entrenadores que desean proporcionar a sus jugadores entrenamientos personalizados y que les acompañe tanto a la hora del partido, como a la hora de la planificación de entrenamientos.

Por tanto, se llegó al acuerdo entre tutores y alumno de desarrollar dicha herramienta en forma de aplicación web multiplataforma, con el objeto de que pueda ser accedida tanto desde fuera de casa, mediante el teléfono móvil o tablet en un partido, como desde el ordenador de sobremesa o portátil a posteriori para analizar resultados.

Fruto de dicho acuerdo, surgieron ciertas ideas para extender la funcionalidad de la aplicación. La primera de ellas fue generalizar la aplicación para todos los deportes de equipo en lugar de particularizarlo para uno solo.

La siguiente, más importante, fue realizar la aplicación para un uso a nivel de club y no a nivel individual. Esto quiere decir que la aplicación debe permitir a un usuario crear un club, definir varios equipos dentro de este, invitar a otros entrenadores a colaborar y, por último, que los resultados puedan ser estudiados y compartidos por todos los entrenadores del club.

## 1.2 Objetivos

El objetivo principal de este proyecto, tal y como se ha expuesto anteriormente, es el de poner a disposición de todos los entrenadores de equipos deportivos que lo deseen, una aplicación web a la que puedan acceder desde todos sus dispositivos, tanto en su casa como fuera de ella.

Ateniéndonos a las características deseadas, la aplicación puede dividirse en tres partes bien diferenciadas:

La primera característica que debe tener la aplicación es la de hacer posible la gestión de un club por múltiples usuarios. Esto es crear un club, invitar a otros usuarios a colaborar en él y definir equipos dentro del mismo.

Tras ello, se debe poder añadir jugadores y eventos que puedan ocurrir durante un partido y que el entrenador o entrenadores consideren relevante para su estudio estadístico posterior, en vista de reforzar o trabajar ciertos aspectos en los entrenamientos.

La segunda parte de la aplicación corresponde a la opción de simular un partido en tiempo real con una interfaz donde el entrenador pueda anotar, por cada unidad de tiempo, el evento que ocurre y el jugador que lo ejecuta.

Por ejemplo, si el deporte es balonmano, uno de los jugadores del equipo es “Juan” y uno de los eventos incluidos por el entrenador es “Pase al pivote”, si el evento tiene lugar en el minuto 7 de partido y es propiciado por dicho jugador, la aplicación debe registrar la ocurrencia de la siguiente forma:

- Minuto: 07:00
- Evento: Pase al pivote.
- Jugador: Juan.

De esta forma, el entrenador irá anotando durante el partido en tiempo real todas las ocurrencias de eventos y sus respectivos jugadores y, cuando finalice el partido, toda la información recogida quedará almacenada en una base de datos, para que puedan servir como entrada a la tercera parte de la aplicación.

La tercera y última parte de la aplicación tomará los datos obtenidos durante los partidos y, mediante ciertos algoritmos estadísticos, será capaz de mostrar gráficos que permitan al entrenador estudiar los resultados, analizarlos y sacar conclusiones.

Gracias a todo lo anterior, el usuario podrá satisfacer el objetivo final del proyecto que es el de planificar entrenamientos personalizados en base a los datos recogidos y analizados por la aplicación.

## 1.3 Tecnologías utilizadas

Para el desarrollo de la aplicación se barajaron varias tecnologías. Al principio, se pensó en desarrollar una aplicación exclusivamente para dispositivos móviles, ya que uno de los objetivos de la aplicación es poderse utilizar de forma remota durante los partidos.

Como se pretendía llegar al mayor número de personas posible, se consideró utilizar el framework Xamarin de Microsoft. Este framework permite, mediante código C#, escribir aplicaciones móviles nativas para sistemas operativos Android e iOS, entre otros, y compartir el código a través de ellas, lo que habría dado lugar a una aplicación móvil multiplataforma.

Tras esto, surgió la idea, ya comentada, de construir la aplicación no solo a nivel de un equipo con un usuario individual, sino a nivel de club con varios usuarios participantes.

Este planteamiento llevó a la reflexión de si la aplicación debería estar disponible también, a parte de para dispositivos móviles, para escritorio. Esto llevó a suponer un posible escenario donde el dueño del club pudiera meterse en la aplicación de escritorio para ver los resultados de los equipos de su club.

Por ello, y para no tener que desarrollar dos aplicaciones diferentes (una para dispositivos móviles y otra para escritorio), se llegó a la conclusión de que lo mejor era confeccionarla en forma de aplicación web.

La ventaja de esto, se pensó, es que una aplicación web puede ser accedida tanto desde dispositivos móviles, como desde dispositivos de sobremesa o escritorio. La aplicación, por tanto, debería ser totalmente responsiva para adaptarse a los distintos tamaños de pantalla, así como ofrecer todas las funcionalidades necesarias para usar en los partidos, a distancia, o bien para usar en casa a la hora de revisar y analizar los resultados obtenidos.

También se meditó la posibilidad de expandir la aplicación, en un futuro y mediante otro Trabajo de Fin de Grado, con el desarrollo de una aplicación móvil nativa que pueda ser usada más fácilmente desde dispositivos móviles, así como de expandir las funcionalidades ya comentadas.

Tras haber decidido que el proyecto tomase la forma de aplicación web, se pasó a determinar la tecnología principal, es decir, el lenguaje de programación. De entrada, se barajó la posibilidad de realizarlo en Java o en Ruby, debido a la experiencia del alumno en dichos lenguajes.

Finalmente, debido a la propuesta del tutor impulsor de la idea y con objeto de poder seguir trabajando en la aplicación de forma posterior, se propuso realizarlo en el lenguaje C#, y más concretamente, con el framework ASP.NET ya que tanto el tutor como su departamento poseen experiencia en dichas tecnologías y podrían seguir expandiendo y manteniendo la aplicación en el futuro.

Uno de los objetivos últimos de la aplicación, a parte de tomar datos durante un partido, es el de analizar dichos datos de forma estadística para obtener gráficos que ayuden al entrenador a tener una idea exacta del rendimiento y la evolución del equipo y de los jugadores para, posteriormente, poder planificar los entrenamientos en base a ellos.

Desde el principio, se estableció que se utilizaría el lenguaje estadístico R para poder utilizar una serie de scripts de análisis que, el tutor impulsor de la idea y experto en el tema, había desarrollado para este fin en un proyecto particular.

Por tanto, se necesitaban integrar de algún modo la aplicación web de ASP.NET y dichos scripts de R. Afortunadamente, para este cometido existe una extensión de .NET llamada R.NET que permite realizar llamadas al lenguaje R dentro de un programa C# y manipular los resultados en la aplicación.

A continuación se detallarán las tecnologías finalmente escogidas para el desarrollo. Se comenzará por una breve descripción de la tecnología para terminar con un razonamiento de por qué ha sido elegida:

- **Visual Studio – Entorno de desarrollo:**

  Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Es compatible con múltiples lenguajes de programación, entre ellos C#, y permite a los desarrolladores crear sitios y aplicaciones web mediante tecnologías como ASP.NET.

  Por las razones anteriormente expuestas, se acabó decidiendo que la aplicación estaría escrita en el lenguaje de programación C# utilizando el framework ASP.NET.

  Como ambos han sido desarrollados por Microsoft, parecía natural utilizar también el entorno de desarrollo más conocido de esta compañía de forma que fuese más eficiente utilizar todas sus tecnologías.

- **C# – Lenguaje de programación para el back-end:**

  C# es un lenguaje de programación orientado a objetos y desarrollado por Microsoft como parte de su plataforma .NET (Anderson, Rick, Documentación de ASP.NET, s.f.).

  Como bien hemos explicado, la elección del lenguaje se debió a la experiencia del tutor y de su departamento en él, ya que ellos debían ser capaces de seguir expandiendo y manteniendo la aplicación en el futuro.

- **Microsoft SQL Server – Base de datos:**

  Microsoft SQL Server es un sistema gestor de base de datos relacional desarrollado por Microsoft.

De inicio, se barajó la posibilidad de utilizar un sistema gestor de base de datos de código abierto como MySQL. Tras ciertas pruebas e intentos de integración de MySQL en entornos y lenguajes de Microsoft como Visual Studio y C#, se llegó a la conclusión de que era más eficiente, intuitivo y sencillo utilizar un sistema gestor de base de datos también desarrollado por Microsoft.

Finalmente se acabó escogiendo Microsoft SQL Server para este cometido.

- **SQL Server Management Studio – Entorno para bases de datos:**

  SQL Server Management Studio es un entorno integrado para administrar cualquier infraestructura de SQL y ha sido desarrollado por Microsoft.

  Como se optó por usar de forma casi exclusiva herramientas desarrolladas por Microsoft, también fue un paso lógico elegir un entorno para bases de datos perteneciente a esta compañía.

- **ASP.NET – Framework de desarrollo de aplicaciones web:**

  ASP.NET es un entorno para aplicaciones web desarrollado por Microsoft y que permite a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework (Anderson, Rick, Documentación de ASP.NET, s.f.).

  El framework para la aplicación web, así como el lenguaje de programación, fueron designados por el tutor, como ya se ha explicado anteriormente, debido a su experiencia en ellos, con objeto de poder seguir expandiendo y manteniendo la aplicación en el futuro.

- **MVC – Framework de implementación para ASP.NET:**

  ASP.NET MVC es un framework para la creación de aplicaciones ASP.NET que se basa en el patrón de diseño Modelo-Vista-Controlador, el cual se usa en programación para separar conceptos y para facilitar la reutilización de código (Anderson, Rick, Documentación de ASP.NET, s.f.).

Este framework fue elegido para el desarrollo del proyecto porque, además de proporcionar un muy bajo acoplamiento entre conceptos, facilita la creación de la aplicación por iteraciones, permitiendo cómodamente la extensión y modificación de código.

- **Identity – Sistema de identificación:**

  ASP.NET Core Identity es un framework que otorga las funciones de registro y de inicio de sesión, entre otras, en sistemas ASP.NET (Anderson, Rick, Documentación de ASP.NET, s.f.).

  Para la autenticación en la aplicación, se ha decidió utilizar este framework por su rapidez de configuración y su robustez. Identity permite, además, utilizar una base de datos para almacenar nombres de usuario, contraseñas y datos de perfil.

- **R – Lenguaje de programación estadístico:**

  R es un lenguaje de programación con enfoque al análisis estadístico (R Documentation and manuals, s.f.).

  Desde el principio del acuerdo para este proyecto, se había establecido el utilizar R para la parte del análisis estadístico de los partidos.

  El tutor ya tenía desarrollados una serie de funciones en R que servían este propósito y la idea siempre había sido utilizar dichas funciones para la aplicación.

- **RMarkdown – Lenguaje de marcado para R:**

  RMarkdown es un lenguaje de marcado que permite procesar código R y mostrar resultados en distintos formatos, incluyendo HTML y PDF (R Markdown, s.f.).

  Con el objetivo de mostrar los resultados del procesamiento estadístico de R de forma automática, se ha utilizado RMarkdown para generar directamente código HTML que pudiera presentar los resultados del análisis al usuario.

- **RStudio – Entorno de desarrollo para R:**

  RStudio es un entorno de desarrollo integrado para R que incluye una consola, editor de código y herramientas gráficas para la ejecución de código R.

  RStudio ha sido utilizado como entorno de desarrollo para el estudio de los scripts de R necesarios para el análisis estadístico de los partidos en la aplicación.

- **R.NET – Extensión de R para .NET:**

  R.NET permite al framework .NET interoperar con el lenguaje estadístico R en el mismo proceso e intercambiar variables y datos para su manipulación en C# o en cualquier lenguaje de .NET (R.NET documentation, s.f.).

  R.NET ha sido necesario para integrar las funciones de R desarrolladas por el profesor y encargadas de realizar el análisis estadístico de los partidos en la aplicación web.

  Esta extensión, ha hecho posible tomar datos de la aplicación web, introducirlos como variables de entrada a funciones escritas en R y extraer los resultados de dichas funciones para su presentación al usuario de vuelta en la aplicación web.

- **HTML – Lenguaje de marcado para el front-end:**

  HTML (HyperText Markup Language) es un lenguaje de marcado para la elaboración de páginas web (W3Schools Online Web Tutorials, s.f).

  Este lenguaje es el estándar absoluto para escribir el front-end de las páginas web y no hay otra alternativa a elegir.

- **CSS – Lenguaje de estilo para el front-end:**

  CSS (Cascading Style Sheets) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado como HTML (W3Schools Online Web Tutorials, s.f).

  Igual que con HTML no hay otra alternativa para el estilo de las páginas web.

- **JavaScript – Lenguaje de programación para el front-end:**

  JavaScript es un lenguaje de programación interpretado orientado a objetos. Se utiliza en la parte del lado del cliente en una aplicación web permitiendo mejoras en la interfaz de usuario, así como páginas web dinámicas (W3Schools Online Web Tutorials, s.f).

  Uno de los requisitos principales de la aplicación, es que los partidos deben ser simulados con un marcador y cronómetro en tiempo real y las páginas web HTML son, por naturaleza, estáticas.

  La única forma de conseguir una página dinámica con las características descritas era utilizando JavaScript como tecnología para alterar los elementos de la página en tiempo real.

- **JQuery – Biblioteca para JavaScript en el front-end:**

  JQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web (JS Foundation, jQuery API Documentation, s.f.).

  Al estar obligados a utilizar JavaScript para realizar los partidos en tiempo real y al ser JQuery una forma más sencilla y cómoda de utilizar JavaScript, se optó por usar directamente esta tecnología para llevar a cabo este tipo de acciones.

- **Bootstrap – Librería para el front-end:**

  Bootstrap es la librería de front-end más popular y utilizada del mundo. Esta librería se vale de las tecnologías HTML, CSS y JavaScript para proporcionar elementos preparados específicamente para que un usuario los pueda usar en su interfaz y que se adapten a todas sus necesidades (Otto, Mark, Thornton, Jacob and Bootstrap contributors, Bootstrap – The world's most popular mobile-first and responsive front-end framework, s.f.).

  Se ha elegido Bootstrap para el desarrollo del proyecto ya que uno de los objetivos principales del mismo es que la interfaz sea usable en distintos tamaños de pantalla, bien sea en dispositivos móviles o bien en ordenadores de sobremesa.

  Bootstrap ofrece elementos para la interfaz que directamente son responsivos y que cumplen esos objetivos sin necesidad de más configuración por parte del usuario.

- **AJAX – Técnica de desarrollo web para el front-end:**

  AJAX (Asynchronous JavaScript And XML) es una técnica de desarrollo web para realizar cambios sobre las páginas web sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones (JS Foundation, jQuery API Documentation, s.f.).

  Como parte de la funcionalidad principal de la aplicación, los partidos en tiempo real deben permitir realizar acciones como registrar un evento, sustituir a un jugador o cambiar de periodo.

  Para todas las acciones mencionadas, se han utilizado las técnicas AJAX con objeto de comunicarnos con el servidor, ejecutar métodos y visualizar los resultados sin tener que recargar la página cada vez, evitando así la pérdida de otros datos relevantes como el marcador o el tiempo de partido.

- **GIT – Software de control de versiones:**

  GIT es un software de control de versiones que realiza un seguimiento de los cambios en el código fuente durante el desarrollo software.

  Con el propósito de mantener un registro de hitos y versiones de la aplicación, se ha hecho uso de la tecnología GIT. Dicha tecnología también nos puede resultar útil en los casos en que una característica de aplicación es deficiente y hay que restaurar el código a una versión anterior con facilidad.

- **SourceTree – Cliente de escritorio para GIT:**

  SourceTree es un Cliente de escritorio para GIT propiedad de la empresa desarrolladora Atlassian.

  Se ha escogido SourceTree con el objetivo de simplificar y hacer más sencillo el uso de GIT mediante una interfaz gráfica, ante la alternativa de usar GIT desde consola de comandos.

- **BitBucket – Repositorio remoto para GIT:**

  BitBucket es un repositorio remoto web para el control de versiones propiedad de la empresa desarrolladora Atlassian.

  Se ha escogido BitBucket como repositorio remoto para las versiones del proyecto debido a la experiencia del alumno con él.

## 1.4 Estructura de la memoria

A continuación, se expondrá la estructura de la memoria, con una breve explicación de cada capítulo:

- **Introducción:**

  En este capítulo se exponen las motivaciones que han llevado a este proyecto, así como sus objetivos principales, definiendo las características de la aplicación y lo que se espera conseguir con ella.

  Por último, se detallan las tecnologías utilizadas durante todo el desarrollo acompañadas por un breve razonamiento de por qué han sido elegidas.

- **Requisitos del proyecto:**

  En este capítulo se hablará de los requisitos del proyecto, definiendo con detalle los requisitos funcionales y no funcionales.

- **Modelado y diseño del sistema:**

  En este punto, se prestará atención tanto al modelo de la base de datos como al diseño de la aplicación y se expondrán las razones que han llevado a ello.

- **Implementación:**

  Debido al uso de una metodología ágil para el desarrollo de la aplicación, se han llevado a cabo una serie de iteraciones completas, que van desde la concepción del programa, pasando por el diseño del mismo y terminando por la implementación.

  Como resultado de las iteraciones se han obtenido varios prototipos plenamente funcionales.

  Por tanto, este apartado describirá las iteraciones ágiles que han tenido lugar durante el desarrollo del proyecto pasando por sus distintas fases.

- **Pruebas:**

  Este capítulo detallará el proceso de pruebas al que ha sido sometida la aplicación, tras finalizar el desarrollo.

- **Conclusiones:**

  Finalmente, se dedicará un capítulo a las conclusiones y resultados que se han obtenido del proceso de desarrollo del trabajo.

  Se hablará de si se han cumplidos los objetivos, de las líneas futuras que pueden surgir del proyecto y de las dificultades que han surgido por el camino.

- **Anexo – Manual de Usuario:**

  Se adjunta a la memoria un anexo que contiene el manual de uso de la aplicación donde se describe cómo realizar cada una de las opciones disponibles en el sistema.

# 2

# Requisitos

En este apartando se describirán la especificación y análisis de requisitos, que estarán divididos en funcionales y no funcionales.

## 2.1 Requisitos funcionales

Los requisitos funcionales nacen del consenso entre alumno y tutor, donde este último actúa como cliente de la aplicación.

Mediante una serie de entrevistas, se han documentado las características últimas y las necesidades que la aplicación debe satisfacer, desde el punto de vista del cliente y usuario final.

Se ha estimado ordenar los requisitos por prioridades debido a que hay una clara distinción entre los más importantes para el cliente y los que no lo son tanto.

A continuación se detallan todos los requisitos funcionales del sistema, proporcionando una breve descripción, su prioridad y sus requisitos derivados:

### 2.1.1 Requisitos relativos a la simulación de partidos

- **Definir un equipo en la aplicación:**
  - Identificador: RF-001
  - Prioridad: Alta.
  - Descripción: La aplicación debe dar soporte al usuario para crear un equipo deportivo dentro de un club.
  - Requisitos derivados:
    - RF-002 – Definir jugadores para un equipo.
    - RF-003 – Definir eventos para un equipo.

- **Definir jugadores para un equipo:**
  - Identificador: RF-002
  - Prioridad: Alta.
  - Descripción: Se debe permitir crear y añadir jugadores a un equipo y que tengan como campos dorsal y nombre.
  - Requisitos derivados:
    - RF-004 – Distinguir entre jugadores titulares y suplentes.
    - RF-010 – Registrar ocurrencias de eventos durante el partido.

- **Definir eventos para un equipo:**
  - Identificador: RF-003
  - Prioridad: Alta.
  - Descripción: Se debe permitir crear y añadir eventos que se consideren relevantes para el análisis posterior a un equipo.
  - Requisitos derivados:
    - RF-010 – Registrar ocurrencias de eventos durante el partido.

- **Distinguir jugadores titulares de suplentes:**
  - Identificador: RF-004

- Prioridad: Alta.
- Descripción: La aplicación debe distinguir entre jugadores titulares y suplentes dentro de un mismo equipo.
- Requisitos derivados:
  - RF-009 – Realizar sustituciones durante los partidos.

- **Simular partidos en tiempo real:**
  - Identificador: RF-005
  - Prioridad: Alta.
  - Descripción: La aplicación debe ser capaz de ofrecer una simulación en tiempo real de partidos deportivos dentro de una temporada concreta.
  - Requisitos derivados:
    - RF-006 – Ofrecer un cronómetro durante el partido.
    - RF-007 – Ofrecer un marcador durante el partido.
    - RF-008 – Cambiar de periodo durante el partido.
    - RF-009 – Realizar sustituciones durante el partido.
    - RF-010 – Registrar ocurrencias de eventos durante el partido.
    - RF-017 – La aplicación debe procesar, mediante algoritmos estadísticos, los resultados de un partido.

- **Ofrecer un cronómetro durante el partido:**
  - Identificador: RF-006
  - Prioridad: Alta.
  - Descripción: Durante el partido, debe mostrarse un cronómetro en tiempo real que indique el periodo del partido y los minutos y segundos transcurridos hasta el momento en dicho periodo.
  - Requisitos derivados:
    - RF-011 – El cronómetro debe poder pausarse y reanudarse.

- **Ofrecer un marcador durante el partido:**
  - Identificador: RF-007
  - Prioridad: Media.
  - Descripción: Durante el partido, debe mostrarse un marcador con el resultado del partido hasta el momento.
  - Requisitos derivados:
    - RF-012 – El marcador debe poder manipularse.

- **Cambiar de periodo durante el partido:**
  - Identificador: RF-008
  - Prioridad: Media.
  - Descripción: Durante el partido, debe poder cambiarse de periodo, suponiendo que haya más de uno, y al hacerlo, el cronómetro debe reiniciarse.
  - Requisitos derivados:
    - Ninguno.

- **Realizar sustituciones durante el partido:**
  - Identificador: RF-009
  - Prioridad: Alta.
  - Descripción: Durante el partido, deben poder sustituirse jugadores suplentes por titulares.
  - Requisitos derivados:
    - RF-013 – La vista del partido debe ofrecer una lista de jugadores titulares y suplentes.

- **Registrar ocurrencias de eventos durante el partido:**
  - Identificador: RF-010
  - Prioridad: Alta.
  - Descripción: Durante el partido, deben poder registrarse las ocurrencias de eventos, así como del jugador que las ejecuta.
  - Requisitos derivados:

        - RF-014 – La vista del partido debe ofrecer una lista de eventos.

- **El cronómetro debe poder pausarse y reanudarse:**
    - Identificador: RF-011
    - Prioridad: Media.
    - Descripción: Durante el partido, el cronómetro que lleva la cuenta del tiempo del periodo debe poder pausarse y reanudarse cuando el usuario lo estime oportuno.
    - Requisitos derivados:
        - Ninguno.

- **El marcador debe poder manipularse:**
    - Identificador: RF-012
    - Prioridad: Media.
    - Descripción: Durante el partido, el marcador que lleva la cuenta del resultado del partido debe poder incrementarse y decrementarse, tanto de un equipo como de otro.
    - Requisitos derivados:
        - Ninguno.

- **La vista del partido debe ofrecer una lista de jugadores titulares y suplentes:**
    - Identificador: RF-013
    - Prioridad: Alta.
    - Descripción: Durante el partido, debe presentarse una vista al usuario con la lista de jugadores titulares y otra lista de los jugadores suplentes, de cara a realizar sustituciones entre ellos.
    - Requisitos derivados:
        - Ninguno.

- **La vista del partido debe ofrecer una lista de eventos:**
  - Identificador: RF-014
  - Prioridad: Alta.
  - Descripción: Durante el partido, debe presentarse una vista al usuario con la lista de los eventos que previamente ha introducido en la aplicación y que considere relevantes para su estudio posterior.
  - Requisitos derivados:
    - Ninguno.

### 2.1.2 Requisitos relativos al análisis estadístico

- **La aplicación debe presentar un análisis estadístico de los resultados de la temporada:**
  - Identificador: RF-015
  - Prioridad: Alta.
  - Descripción: La aplicación debe presentar una serie de gráficos, computados mediante algoritmos estadísticos, que permitan al usuario valorar la evolución de sus jugadores a lo largo de la temporada en función de los eventos que dicho usuario haya considerado relevantes para tal fin.
  - Requisitos derivados:
    - RF-016 – La aplicación debe integrar algoritmos de análisis realizados con el lenguaje estadístico R.
    - RF-017 – La aplicación debe procesar, mediante algoritmos estadísticos, los resultados de un partido.

- **La aplicación debe integrar algoritmos de análisis realizados con el lenguaje estadístico R:**
  - Identificador: RF-016
  - Prioridad: Alta.

- Descripción: La aplicación integrará algoritmos escritos en el lenguaje R y realizados por el tutor que serán los que procesarán la información de los partidos, realizarán el análisis estadístico y presentarán los gráficos al usuario para su estudio.
- Requisitos derivados:
  - Ninguno.

- **La aplicación debe procesar, mediante algoritmos estadísticos, los resultados de un partido:**
  - Identificador: RF-017
  - Prioridad: Alta.
  - Descripción: Al finalizar un partido, los algoritmos de análisis escritos en el lenguaje R e integrados en la aplicación, tomarán los resultados del mismo y, en conjunción con los resultados del resto de la temporada, realizarán un análisis estadístico con el fin de presentar los resultados al usuario.
  - Requisitos derivados:
    - Ninguno.

### 2.1.3 Requisitos relativos a la gestión de un club deportivo

- **La aplicación debe permitir el registro de usuarios:**
  - Identificador: RF-018
  - Prioridad: Alta.
  - Descripción: Un usuario debe poder registrarse en la aplicación, mediante una cuenta de e-mail y una contraseña.
  - Requisitos derivados:
    - RF-019 – La aplicación debe permitir la autenticación de usuarios.

- **La aplicación debe permitir la autenticación de usuarios:**
  - Identificador: RF-019
  - Prioridad: Alta.
  - Descripción: Un usuario debe poder autenticarse en la aplicación mediante una cuenta de e-mail y una contraseña, previo registro, para poder acceder a sus datos.
  - Requisitos derivados:
    - Ninguno.

- **Los usuarios podrán crear un club deportivo:**
  - Identificador: RF-020
  - Prioridad: Alta.
  - Descripción: Un usuario registrado en la aplicación podrá definir en ella uno o varios clubes deportivos, los cuales servirán de plataforma para gestionar sus datos. Tras crear un club, los usuarios podrán definir en él temporadas y equipos y, en estos últimos, añadir jugadores y eventos.
  - Requisitos derivados:
    - RF-001 – Definir un equipo en la aplicación.
    - RF-021 – Los usuarios de un club podrán invitar a otros usuarios de la aplicación a formar parte del club.
    - RF-023 – Los usuarios podrán definir temporadas en un club.

- **Los usuarios de un club podrán invitar a otros usuarios de la aplicación a formar parte del club:**
  - Identificador: RF-021
  - Prioridad: Media.
  - Descripción: Ya que la aplicación está pensada a nivel de club deportivo, se ha desarrollado con la idea de que pueda haber, dentro de un mismo club, varios equipos (por ejemplo correspondientes a las distintas categorías), con varios entrenadores gestionando al mismo tiempo.

  - Requisitos derivados:
    - RF-022 – Los usuarios se pueden invitar a un club como administradores o como entrenadores.

- **Los usuarios se pueden invitar a un club como administradores o como entrenadores:**
  - Identificador: RF-022
  - Prioridad: Baja.
  - Descripción: Un usuario podrá ser invitado a formar parte de un club de dos formas, administrador o entrenador. Los primeros tendrán ciertos privilegios que los segundos no, como por ejemplo la invitación de otros usuarios al club o la creación y eliminación de equipos en el club.
  - Requisitos derivados:
    - Ninguno.

- **Los usuarios podrán definir temporadas en un club:**
  - Identificador: RF-023
  - Prioridad: Alta.
  - Descripción: Un usuario de un club podrá definir, para el mismo, distintas temporadas en las cuales podrá simular partidos para después obtener el análisis estadístico resultante.
  - Requisitos derivados:
    - RF-005 – Simular partidos en tiempo real.

## 2.2 Requisitos no funcionales

Los requisitos no funcionales no están explícitamente definidos por el cliente, no obstante, son necesarios para que la aplicación funcione de la forma deseada.

En la práctica, son primordiales para el éxito del proyecto y, aunque son difíciles de determinar, hacerlo debe ser una pieza clave del desarrollo. Ellos son los encargados de imponer restricciones en el diseño o la implementación y corresponden a propiedades o cualidades que el producto debe tener.

A continuación, se detallarán los requisitos no funcionales que se han podido identificar en el sistema, divididos en categorías que corresponden a aquellos aspectos de la aplicación a los que aluden.

### 2.2.1 Requisitos de aspecto

- **La aplicación tendrá una interfaz estéticamente agradable para el usuario:**
  - Identificador: RNF-001
  - Prioridad: Media.
  - Descripción: La aplicación debe resultar atractiva visualmente para el usuario. Se pondrá especial atención a la paleta de colores utilizada.
  - Requisitos derivados:
    - Ninguno.

- **La aplicación tendrá una vista diferente para cada tarea:**
  - Identificador: RNF-002
  - Prioridad: Media.
  - Descripción: Con el fin de abrumar lo menos posible al usuario, la interfaz deberá ofrecer vistas simples, que sirvan solo para una tarea concreta.
  - Requisitos derivados:
    - Ninguno.

- **La aplicación se adaptará al tamaño de pantalla de distintos dispositivos:**
    - Identificador: RNF-003
    - Prioridad: Alta.
    - Descripción: La aplicación está pensada para usarse tanto desde ordenadores de escritorio como desde dispositivos móviles. Por tanto, la interfaz debe ser responsiva y se debe adaptar a los distintos tamaños de pantalla sin perder información y usabilidad.
    - Requisitos derivados:
        - RF-004 – La aplicación deberá ser intuitiva y fácil de usar para el usuario.

### 2.2.2 Requisitos de usabilidad

- **La aplicación deberá ser intuitiva y fácil de usar para el usuario:**
    - Identificador: RNF-004
    - Prioridad: Media.
    - Descripción: En vista de poder ser usada por usuarios con poco o ningún conocimiento técnico, la interfaz de la aplicación debe ser lo más intuitiva, fácil de usar y simple posible desde cualquier dispositivo.
    - Requisitos derivados:
        - RF-005 – El número de clics para realizar cualquier acción debe ser mínimo.

- **El número de clics para realizar cualquier acción deberá ser mínimo:**
    - Identificador: RNF-005
    - Prioridad: Media.
    - Descripción: Se debe intentar no hacer tedioso el uso de la aplicación. Por tanto se procurará reducir el número de clics

necesarios para realizar cualquier acción en la medida de lo posible.

- Requisitos derivados:
  - Ninguno.

### 2.2.3 Requisitos de seguridad

- **La aplicación no debe permitir a un usuario ver los datos de otro usuario:**
  - Identificador: RNF-006
  - Prioridad: Alta.
  - Descripción: El registro y autenticación de un usuario en la aplicación, posibilitará al mismo el acceso a sus datos y solo a sus datos. Bajo ningún concepto un usuario tendrá acceso a los datos de otro usuario (clubes, equipos, jugadores...). Al ser esta una aplicación web hay que prestar especial atención al acceso a los datos por URL.
  - Requisitos derivados:
    - RF-007 – La aplicación no debe permitir a un usuario modificar los datos de otro usuario.
    - RF-008 – La aplicación deberá estar protegida de la falsificación de peticiones en sitios cruzados.

- **La aplicación no debe permitir a un usuario modificar los datos de otro usuario:**
  - Identificador: RNF-007
  - Prioridad: Alta.
  - Descripción: Un usuario no podrá ver y, por tanto, no podrá modificar en ningún caso, los datos de otro usuario de la aplicación. Al ser esta una aplicación web hay que prestar especial atención al acceso a los datos por URL.
  - Requisitos derivados:

        - RF-008 – La aplicación deberá estar protegida de la falsificación de peticiones en sitios cruzados.

- **La aplicación deberá estar protegida contra la falsificación de peticiones en sitios cruzados:**
    - Identificador: RNF-008
    - Prioridad: Alta.
    - Descripción: La falsificación de peticiones en sitios cruzados o cross-site request forgery es una vulnerabilidad que consiste en falsificar peticiones de usuarios no autorizados con el fin de modificar o acceder a datos de la aplicación desde otro sitio web.

      Como se ha expuesto anteriormente, la aplicación no debe permitir, ni a otro usuario ni a nadie, acceder o modificar de ninguna manera los datos de otro usuario.
    - Requisitos derivados:
        - Ninguno.

- **La aplicación deberá estar protegida contra la inyección de SQL:**
    - Identificador: RNF-009
    - Prioridad: Alta.
    - Descripción: La inyección de SQL es una vulnerabilidad en el nivel de validación que permite a un intruso realizar operaciones no autorizadas en una base de datos.

      La aplicación, por tanto, deberá usar sentencias SQL parametrizadas y validadas para protegerse de este problema de seguridad.
    - Requisitos derivados:
        - Ninguno.

### 2.2.4 Requisitos de rendimiento

- **El tiempo de respuesta de la aplicación debe ser mínimo:**
  - Identificador: RNF-010
  - Prioridad: Alta.
  - Descripción: Tanto la implementación como el diseño de la aplicación se deben enfocar a realizar un producto que sea lo más óptimo posible en cuanto a rendimiento y tiempo de respuesta.
  - Requisitos derivados:
    - Ninguno.

### 2.2.5 Requisitos de disponibilidad

- **Se debe poder acceder a la aplicación desde cualquier dispositivo con conexión a internet:**
  - Identificador: RNF-011
  - Prioridad: Alta.
  - Descripción: La aplicación debe desarrollarse de forma en que pueda ser usada desde ordenadores de escritorio o de sobremesa y desde teléfonos móviles o tabletas.
  - Requisitos derivados:
    - RF-012 – La aplicación debe estar disponible para ser accedida por cualquier usuario en cualquier momento.

- **La aplicación debe estar disponible para ser accedida por cualquier usuario en cualquier momento:**
  - Identificador: RNF-012
  - Prioridad: Alta.
  - Descripción: La aplicación está pensada para soportar múltiples usuarios al mismo tiempo. Estos deberán tener la

posibilidad de acceder a ella en cualquier momento y desde cualquier dispositivo con conexión a internet.

- Requisitos derivados:
  - Ninguno.

### 2.2.6 Requisitos de documentación

- **La aplicación debe disponer de un manual de usuario:**
  - Identificador: RNF-013
  - Prioridad: Alta.
  - Descripción: En vista de que todo usuario de la aplicación pueda entender los entresijos de la misma, se pondrá a su disposición un manual de usuario que relatará con detalle todas las funcionalidades contenidas en ella.
  - Requisitos derivados:
    - Ninguno.

# 3

# Modelado y diseño del sistema

En este capítulo se hablará sobre el proceso de diseño del sistema. En primer lugar, se podrá ver cuál ha sido el diseño final de la base de datos acompañado de una explicación de cada entidad y de qué supone cada una para la aplicación final.

También se prestará atención al diseño de la aplicación en sí. Se examinará qué patrón de diseño se ha elegido para construirla y cómo ha sido utilizado. Finalmente, se podrá observar un ejemplo concreto del diseño de una entidad real del sistema.

## 3.1 Modelado y diseño de la base de datos

Pese a que la base de datos se ha ido diseñando y modelando de forma iterativa, a continuación se presentará el diagrama completo del sistema. Tras ello, se describirán cada una de las entidades detallando su finalidad dentro de la aplicación.

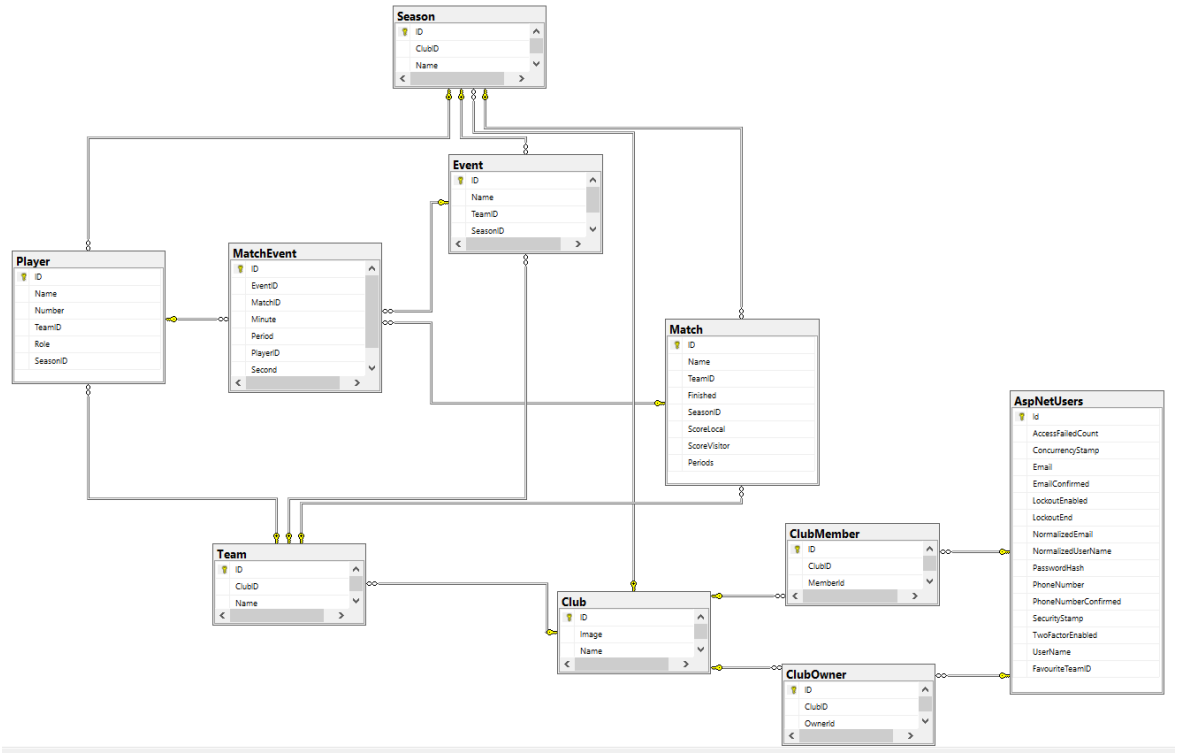


*Figura 1. Modelo de la Base de Datos.*

Las entidades que han sido necesarias para el correcto desarrollo de la aplicación son, como puede verse en la Figura 1, las siguientes:

- **Users:**

O AspNetUsers como viene indicado en la Figura 1 debido a que es el nombre que la tecnología Identity pone por defecto a la clase principal de usuario de la aplicación.

Esta es la entidad que representa a un usuario registrado de la aplicación y que Identity genera automáticamente con todos los campos por defecto que podamos necesitar.

El usuario de la aplicación va  a poder gestionar clubs deportivos y todo lo relacionado con ellos. Por tanto, la entidad AspNetUsers se relaciona con la entidad Club mediante un par entidades intermedias, que representan si un

usuario es un dueño o administrador de un club o bien si es solamente un miembro o entrenador normal del club.

- **Club:**

Club es la entidad que define a los clubes deportivos dentro de la aplicación y actúa como “contenedor” de todas las demás entidades relacionadas. Como hemos dicho, está relacionada con AspNetUsers a través de dos entidades intermedias.

También se relaciona con Season y Team a través de una relación uno a muchos, con lo que se ha querido indicar que dentro de un club se pueden definir varias temporadas y equipos.

- **ClubMember:**

Entidad intermedia que define la relación de muchos a muchos entre AspNetUsers y Club, indicando que un usuario puede ser miembro o entrenador de muchos clubs y que un club puede tener muchos miembros.

Un miembro o entrenador normal del club, es un usuario que participa en el club, es decir, puede realizar ciertas acciones como gestionar un equipo y simular un partido, pero que no tiene ciertos privilegios que un dueño o administrador tendría, como la creación de temporadas y equipos o el borrado de ciertas entidades.

- **ClubOwner:**

La otra entidad intermedia que define la relación de muchos a muchos entre AspNetUsers y Club, indicando que un usuario puede ser dueño o administrador de muchos clubs y que un club puede tener muchos administradores.

Un dueño o administrador de un club posee ciertos privilegios que un miembro normal no tendría. Por ejemplo, los administradores podrán crear y borrar ciertas entidades que el miembro no podría.

- **Season:**

Entidad que representa a las temporadas que están dentro de un club. Una temporada solo pertenece a un club, pero un club puede tener múltiples temporadas.

Las temporadas se han creado como etiqueta para englobar ciertos partidos para que, tras el análisis estadístico, sirvan como guía al entrenador para ver la evolución de sus jugadores a lo largo de ella.

También engloban jugadores y eventos ya que se considera que los jugadores de un equipo, así como los eventos que se consideren relevantes, no serán los mismo de una temporada a otra.

Se ha decidido crear las temporadas a nivel de club en lugar de a nivel de equipo para que haya una sensación de unidad en todos los equipos del club y para que participen todos en la misma temporada.

- **Team:**

Esta entidad representa a los equipos que están dentro de un club. Un club deportivo podrá tener varios equipos, correspondientes a sus distintas categorías, pero un equipo tan solo podrá pertenecer a un club.

El equipo a su vez es el contenedor de jugadores y eventos, así como de distintos partidos que jugará a lo largo de distintas temporadas.

- **Player:**

Entidad que representa a los jugadores que pertenecen a un equipo en una temporada concreta.

Un equipo contará con varios jugadores pero los jugadores solo pueden pertenecer a un equipo.

A su vez, un jugador podrá distinguirse entre suplente y titular dentro de un equipo. Los jugadores participarán en partidos donde se podrán ir anotando los eventos que realizan para, posteriormente, analizar su evolución a lo largo de la temporada.

- **Event:**

Entidad que representa a los eventos que pertenecen a un equipo en una temporada concreta. Un evento pertenecerá a un solo equipo pero un equipo podrá tener muchos eventos.

El usuario de la aplicación (cualquiera que sea administrador o entrenador) podrá definir eventos que considere relevantes dentro de un equipo. En la simulación de un partido, el usuario podrá anotar ocurrencias de eventos para, posteriormente, realizar un análisis estadístico de dichas ocurrencias con el fin de valorar la evolución de los jugadores a lo largo de la temporada.

- **Match:**

Esta entidad representa los partidos de un equipo en una temporada. Una temporada, así como un equipo, podrán tener muchos partidos, pero un partido solo debe pertenecer a una temporada y a un equipo.

Como se ha explicado anteriormente, al simular un partido los usuarios podrán anotar las ocurrencias de eventos por jugador para, posteriormente, realizar un análisis estadístico de con el fin de valorar la evolución de los jugadores a lo largo de la temporada.

Por tanto un partido posee múltiples instancias de la entidad MatchEvent que es la que representa las ocurrencias de eventos en la aplicación.

- **MatchEvent:**

Entidad que representa las ocurrencias de eventos durante un partido. Un partido, por tanto, podrá tener muchas ocurrencias de eventos pero las ocurrencias solo pertenecerán a un partido.

Tras introducir los jugadores y los eventos que se consideren relevantes en el equipo, esta entidad permite registrar en la aplicación si una ocurrencia de un evento por parte de un jugador tiene lugar durante un partido.

La entidad toma todos los datos de la ocurrencia que puedan considerarse relevantes para el posterior análisis estadístico tales como jugador, evento, periodo de partido, minuto y segundo.

Dichos datos serán recogidos por el algoritmo de análisis escrito en R y los añadirá a estadísticas previas de partidos anteriores de la temporada con el fin de que el usuario pueda analizar el rendimiento de sus jugadores a lo largo de ella, así como planificar entrenamientos teniendo en cuenta los resultados.

## 3.2 Modelado y diseño del sistema

Para el diseño del sistema, se ha utilizado el patrón de diseño Modelo-Vista-Controlador. Para ello, se ha contado con la ayuda del framework ASP.NET MVC que facilita crear aplicaciones web ASP.NET utilizando este patrón.

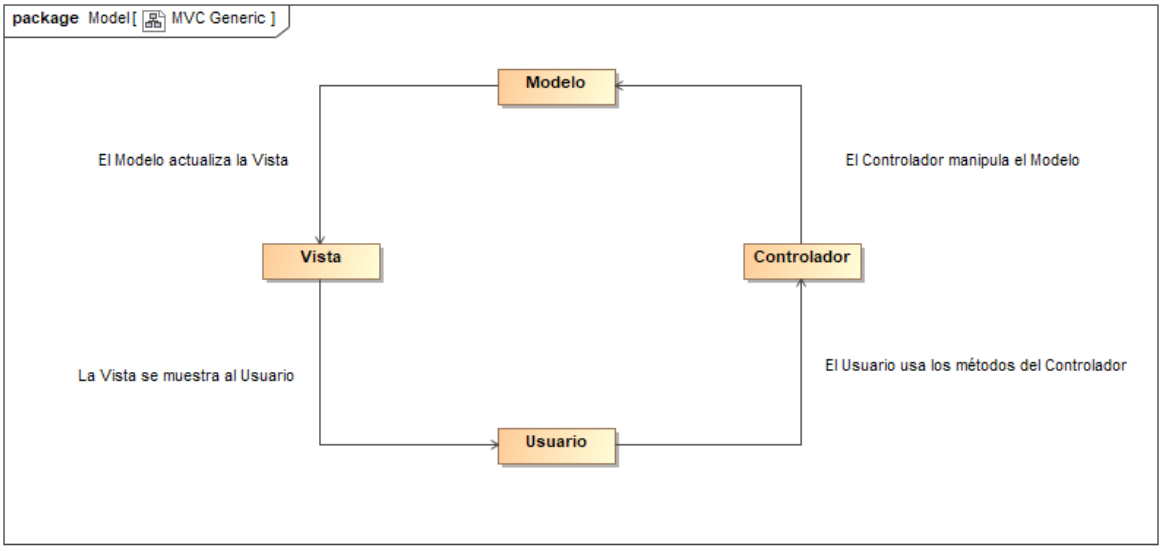


*Figura 2. Patrón MVC.*

Gracias a ello se ha obtenido un diseño desacoplado, reusable, legible y fácil de entender. Este hecho ha sido particularmente útil a la hora de realizar las distintas iteraciones del producto, ya que el patrón facilita la extensión de código de forma cómoda.

En este apartado se detallará cuál ha sido la forma de aplicar el patrón Modelo-Vista-Controlador dentro del contexto de la aplicación. Cabe destacar que la forma de proceder ha sido la misma para todas las entidades del sistema.

Como se ha dicho anteriormente, el framework ASP.NET MVC facilita el desarrollo de aplicaciones web por iteraciones, por lo que no es necesario diseñar todo el sistema (ni siquiera las entidades necesarias en la base de datos) antes de desarrollar la aplicación.

Por tanto, se puede empezar el desarrollo creando una sola entidad o modelo, con un solo controlador y una serie de vistas que, en conjunto, bien pueden corresponder a una aplicación independiente que hace las veces de prototipo.

La manipulación de los datos de una entidad (lectura, actualización, creación y borrado), en consecuencia, quedan a cargo de un y solo un controlador, que es el controlador asociado a esa entidad.

En la Figura 3 se muestra el diagrama de clases que corresponde al diseño Modelo-Vista-Controlador de una de las entidades del sistema. El diseño de la figura representa el diseño básico e inicial del patrón para una entidad cualquiera del sistema.

Para este ejemplo concreto se ha tomado la entidad Club pero el diseño es análogo para todas las entidades.

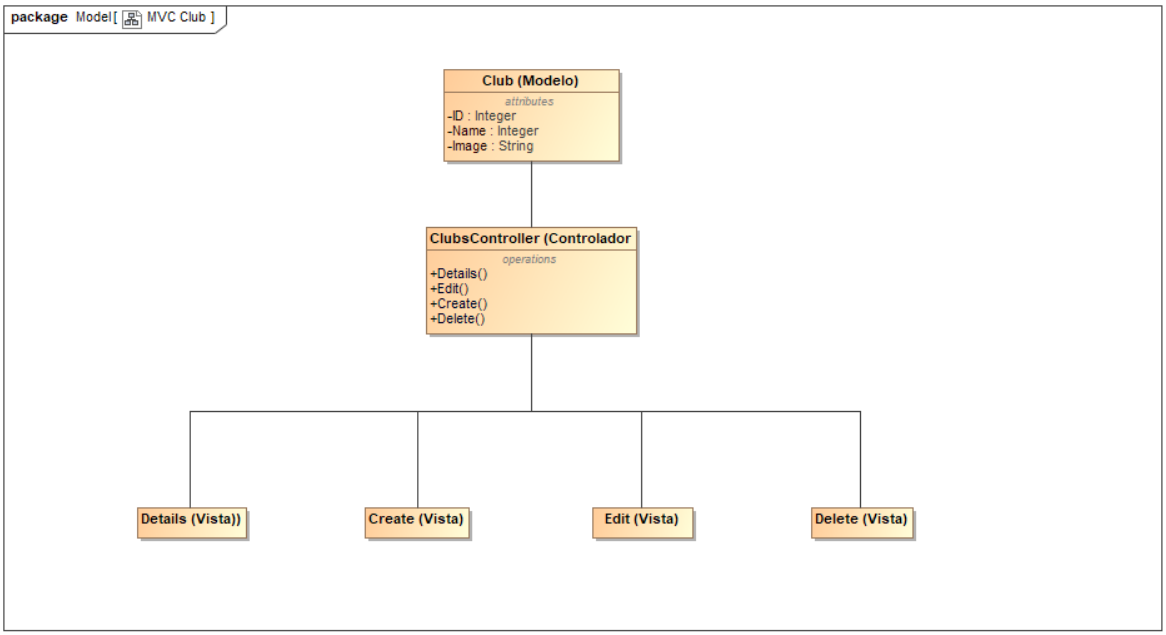


*Figura 3. Club MVC.*

Como se puede ver en la Figura 3, la entidad Club de la base de datos, corresponderá a una clase modelo en la aplicación que tendrá un controlador asociado (el ClubsController o el controlador de clubs).

El controlador de clubs facilita una serie de métodos que son los que se encargan de la creación, borrado, actualización, lectura y, en general, la manipulación de la entidad Club.

Asimismo, cada método del controlador está asociado a una vista de la entidad. Por ejemplo el método Create, que sirve para crear un club, está

asociado a una vista que corresponde a un formulario web donde pueden introducirse los datos necesarios para crear un club en el sistema.

Como se ha explicado anteriormente, este diseño es análogo para todas las entidades del sistema. De esta forma y siguiendo este patrón, se han creado todas las entidades, acciones y vistas que dan forma a la aplicación final y que han hecho posible el desarrollo de todas las funcionalidades especificadas en los requisitos.

La secuencia de intercambio de mensajes entre elementos del sistema también es común para todas las acciones del sistema.

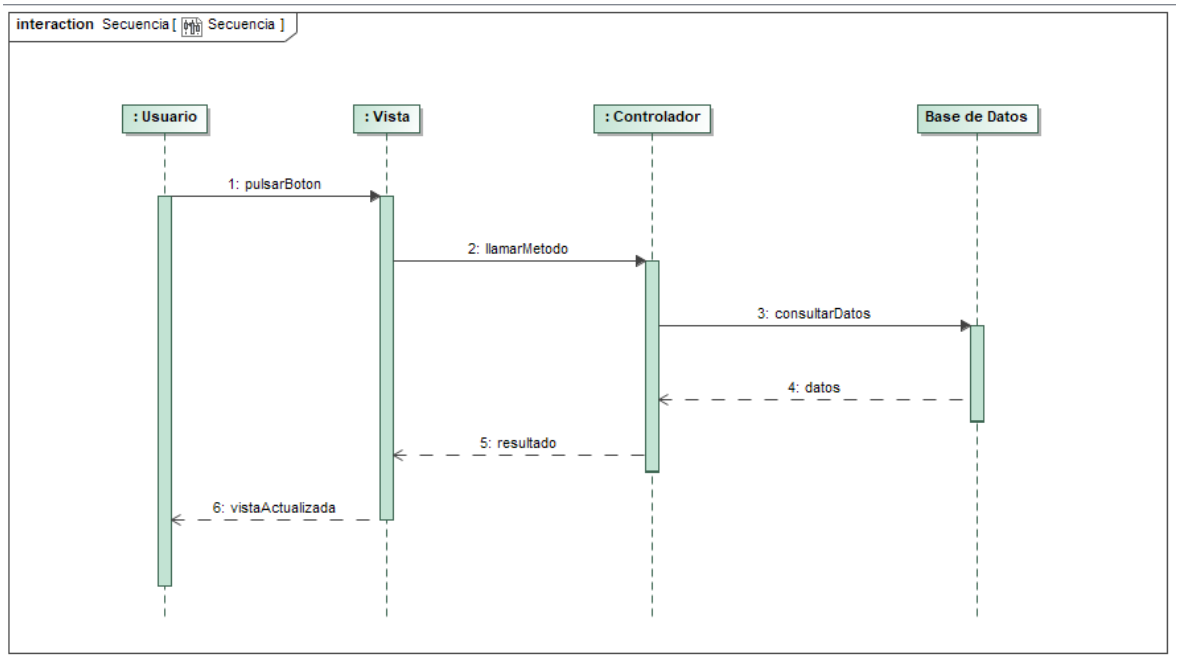


*Figura 4. Diagrama de Secuencia.*

La Figura 4 corresponde a dicho intercambio de mensajes genérico y que se puede extrapolar a todo el sistema. Cuando el usuario pulsa en cualquier botón de una vista, esta llama a un método del controlador que realiza una consulta a la base de datos.

Tras ello, la base de datos devuelve los datos que se le hayan pedido al controlador, que los manipula y actualiza la vista para que el usuario pueda ver el resultado de su acción.

# 4

# Implementación

En este apartado se detallarán los pormenores de la implementación, la cual ha sido desarrollada principalmente en tres iteraciones completas que podrían corresponder a tres prototipos funcionales e independientes.

La primera iteración, corresponde a la parte de gestión de usuarios, clubes y todo lo relativo a ellos. La segunda iteración es la relativa a la simulación de partidos. Y la tercera y última, es la que gestiona la integración con el lenguaje R y la que se encarga de realizar el análisis estadístico de los partidos.

## 4.1 Primera iteración

Como se ha explicado en el capítulo anterior, para el diseño e implementación de la aplicación se ha hecho uso del framework ASP.NET MVC, que facilita el desarrollo de una aplicación web mediante el patrón Modelo-Vista-Controlador.

Este framework proporciona herramientas que permiten, entre otras cosas, construir la aplicación por iteraciones de forma muy sencilla, sin tener que realizar todo el diseño previo a la implementación.

Además, ASP.NET MVC cuenta con un sistema de migraciones que posibilita añadir cualquier modelo o clase que creemos en la aplicación a la base de datos de forma automática, incluso con sus relaciones.

La forma de proceder, por tanto, ha sido la de crear poco a poco los distintos modelos del sistema, añadir su correspondiente entidad a la base de datos, asignar un controlador y, por último, crear las vistas.

La primera iteración del desarrollo ha sido la encargada de definir la estructura de usuarios y clubes dentro del sistema. Al final de esta iteración, se ha obtenido un prototipo donde un usuario podía registrarse en la aplicación, crear un club deportivo, invitar otros usuarios al club, crear temporadas, definir equipos dentro del club y asignar jugadores y eventos al equipo.

### 4.1.1 Gestión de Usuarios

Lo primero que se implementó fue el sistema de gestión de usuarios. Para ello, se ha hecho uso del framework Identity de ASP.NET. El framework crea por defecto y de forma automática las clases necesarias para recrear un sistema de registro y autenticación de usuarios en la aplicación.

Además, una vez creadas las clases, Identity crea una migración que, al ejecutarla, genera todas las entidades necesarias para la gestión de usuarios también en la base de datos con todos sus campos y relaciones.

Una vez generadas todas las clases lo único que tenemos que hacer, si queremos un usuario que se beneficie de todas las funciones de Identity, es crear una clase usuario propia que herede de la clase IdentityUser.

```
namespace CoachJournalWeb.Models
{
    // Add profile data for application users by adding properties to the ApplicationUser class
    public class ApplicationUser : IdentityUser
    {
        public int FavouriteTeamID { get; set; }
        public ICollection<ClubOwner> Ownerships { get; set; }
        public ICollection<ClubMember> Memberships { get; set; }
    }
}
```

*Figura 5. Clase Usuario.*

Una vez hecho esto, nuestro usuario podrá hacer uso del controlador de gestión de usuarios generado por Identity y llamado AccountController. Este controlador proporciona métodos para realizar el registro y la autenticación de usuarios en la aplicación.

El framework también nos proporciona otro controlador, el ManageController, que sirve para administrar nuestra cuenta y que contiene métodos para cambiar los datos relativos a ella como la contraseña o el correo electrónico.

Por último, Identity genera una serie de vistas por defecto que se han adaptado e integrado en la aplicación y que nos permiten realizar la autenticación, el registro y demás funciones comunicándonos con los controladores.

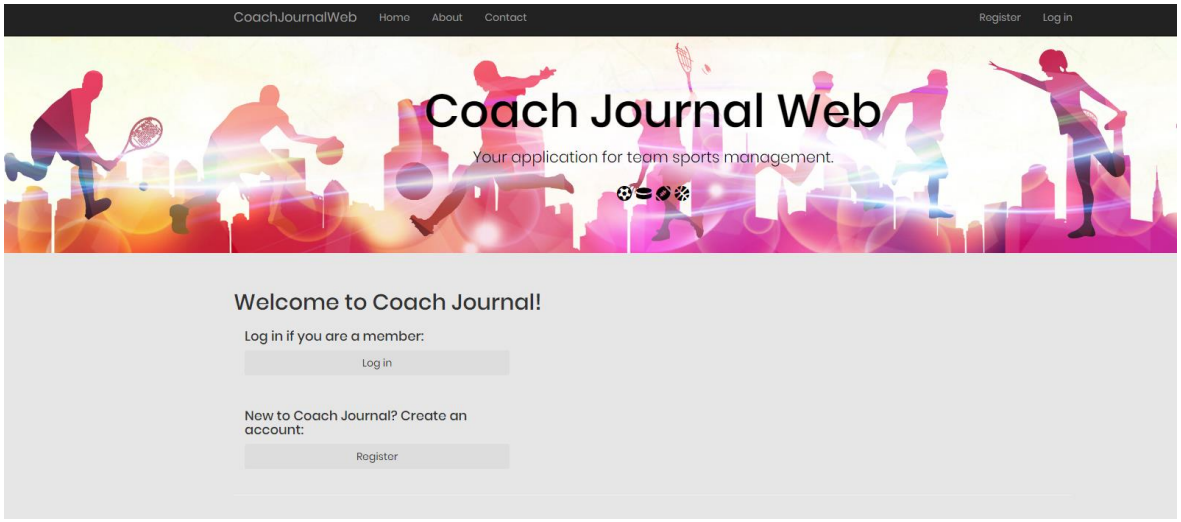


*Figura 6. Vista de Inicio.*

Tras realizar todo esto, se ha obtenido un sistema funcional de gestión de usuarios con todos los modelos, controladores y vistas necesarios para realizar funciones tan importantes como el registro y la autenticación.

### 4.1.2 Gestión de Clubes

Tras tener el sistema de usuarios en marcha, pasamos a la siguiente fase, la de creación y gestión de clubes deportivos.

El primer paso es crear el modelo Club en nuestra aplicación de ASP.NET. El Club es el que engloba todo lo demás que necesitamos para alcanzar los objetivos de la aplicación, es decir, equipos, jugadores, eventos, temporadas y partidos.

```
namespace CoachJournalWeb.Models
{
    public class Club
    {
        public int ID { get; set; }

        [Required]
        public string Name { get; set; }

        public string Image { get; set; }
        public ICollection<ClubOwner> Owners { get; set; }
        public ICollection<ClubMember> Members { get; set; }
        public ICollection<Season> Seasons { get; set; }
        public ICollection<Team> Teams { get; set; }
    }
}
```

*Figura 7. Clase Club.*

En la Figura 7 podemos ver la clase Club dentro de la aplicación. Lo que vemos es el modelo final, ya que vienen especificadas las relaciones con otras clases como equipos y temporadas que, en ese momento, todavía no habían sido creadas.

Lo que sí se construyó en ese momento son sus relaciones con las clases ClubOwner y ClubMember. Estas clases sirven como entidades intermedias entre los usuarios y el club en una relación de muchos a muchos.

Las clases fueron creadas para representar a dos tipos de usuarios dentro de un equipo. El primer tipo son los usuarios administradores o dueños del equipo, que tienen más privilegios y pueden realizar más acciones, y el segundo tipo son los miembros o entrenadores de un equipo, que pueden administrar jugadores y eventos dentro del mismo, así como simular partidos, pero no pueden participar en la manipulación de otras entidades.

```
namespace CoachJournalWeb.Models
{
    public class ClubMember
    {
        public int ID { get; set; }
        public Club Club { get; set; }
        public ApplicationUser Member { get; set; }
    }
}
```

*Figura 8. Clase ClubMember.*

Con ello, los usuarios y los clubes quedan unidos a través de las clases ClubOwner y ClubMember. Tras crear dichas clases, se realizó una migración a la base de datos que generó las tablas y relaciones necesarias para que esto fuese posible.

Tras esto, se crearon las vistas necesarias para que los usuarios pudieran crear, actualizar, borrar y ver uno o varios clubes de su propiedad. En la Figura 9 podemos ver cómo es la vista relativa a los clubes que un usuario administra, en caso de tener varios.

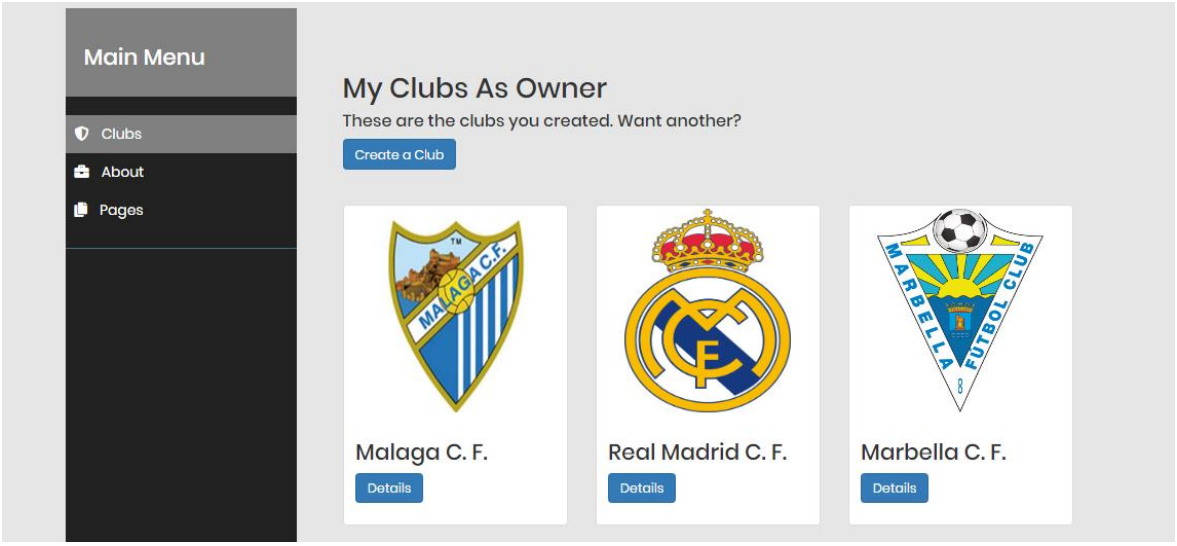


*Figura 9. Lista de Clubes.*

Cabe destacar que se implementaron métodos para que los usuarios pudieran subir una foto del escudo de su club, para mayor personalización de éste. Para ello, se creó una clase ImageUploader que sube una imagen a gusto del usuario al servidor y guarda la ruta en la base de datos como atributo del club.

Después fue necesario crear métodos en el controlador de clubes, así como las vistas necesarias para invitar a otros usuarios a colaborar al club, bien como administradores o bien como miembros.

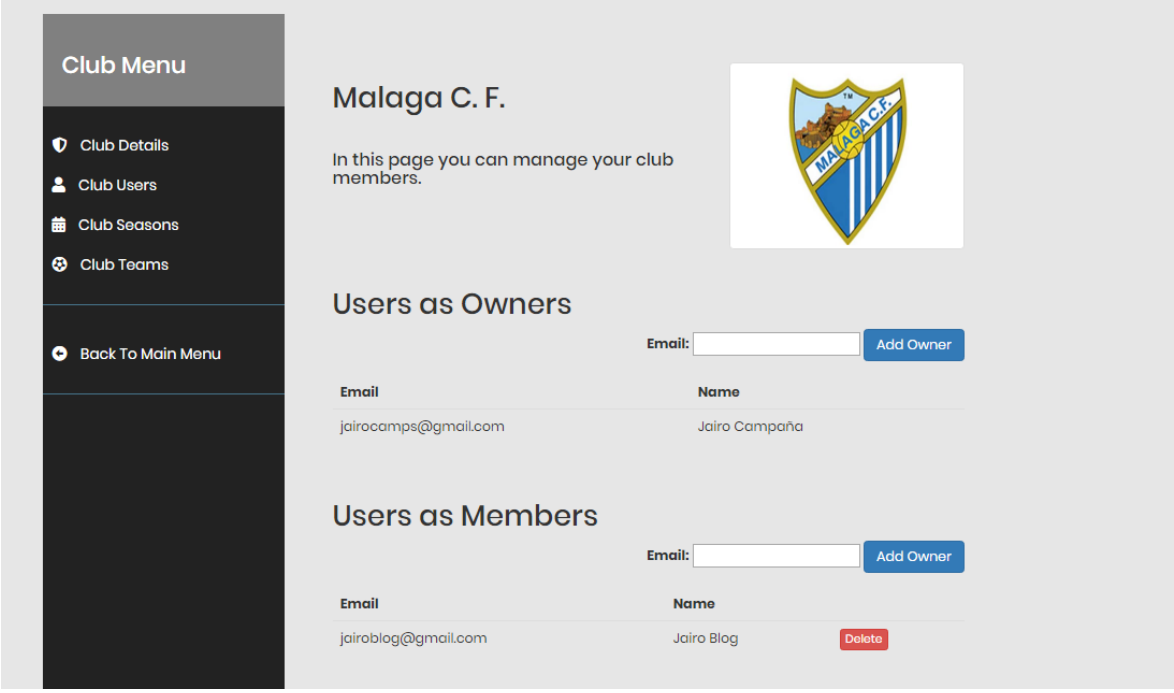


*Figura 10. Usuarios de Club.*

En la Figura 10 podemos ver el resultado de crear dicha vista y métodos. Un usuario que cree un club o que sea administrador de este podrá invitar a otro usuario introduciendo su correo en el apartado correspondiente a miembros, si quiere invitarle como miembro o bien correspondiente a dueño, si quiere invitarle a ser administrador.

Una vez hecho todo esto, contamos con un sistema donde un usuario puede registrarse, crear, ver los detalles, editar o borrar un club e invitar otros usuarios a colaborar en él.

El siguiente paso que se realizó fue el de definir temporadas dentro de un club. La decisión de definir temporadas a nivel de club y no a nivel de equipo fue tomada para incrementar la sensación de unidad en los equipos de un club determinado y que todos pudieran participar en la misma temporada.

Por tanto, a continuación se creó la clase temporada o Season, se establecieron las relaciones con el club y se realizaron las migraciones necesarias a la base de datos. Tras ello, se creó el controlador para las temporadas (SeasonsController) y las vistas necesarias para su gestión.

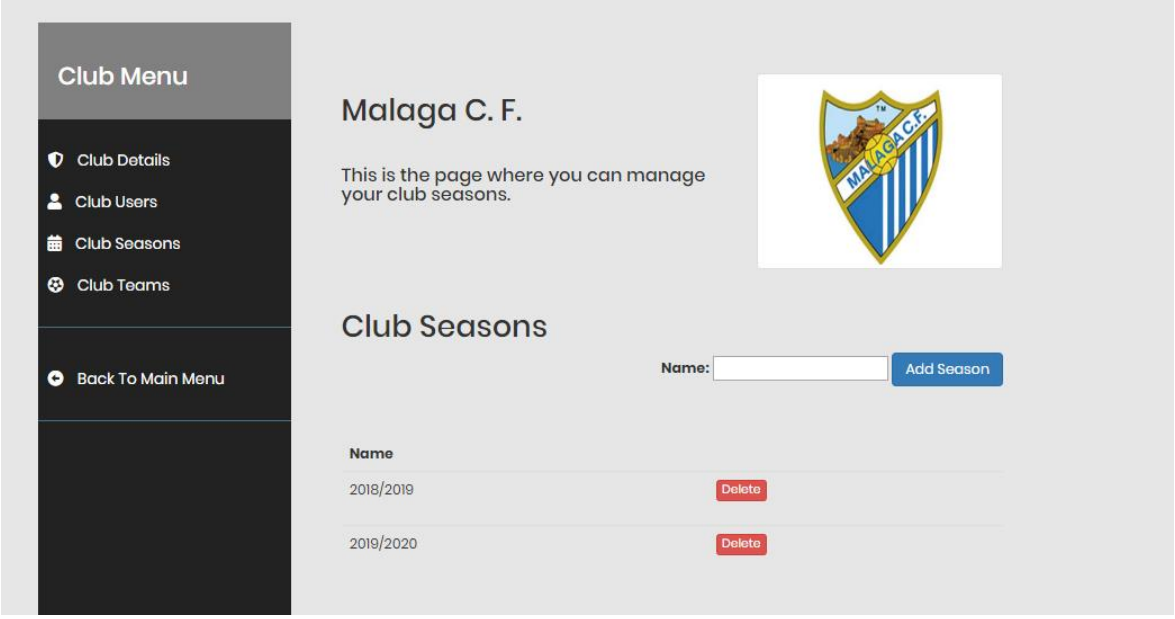


*Figura 11. Temporadas de Club.*

Después, llegó el turno de crear los equipos dentro del club, los cuales corresponden a sus distintas categorías. La forma de proceder fue la misma que con todas las demás entidades. Primero se genera la clase equipo, con todas las relaciones necesarias (pertenece a un club), después se realiza la migración a la

base de datos y por último, se genera el controlador con los métodos y las vistas necesarias para su administración.

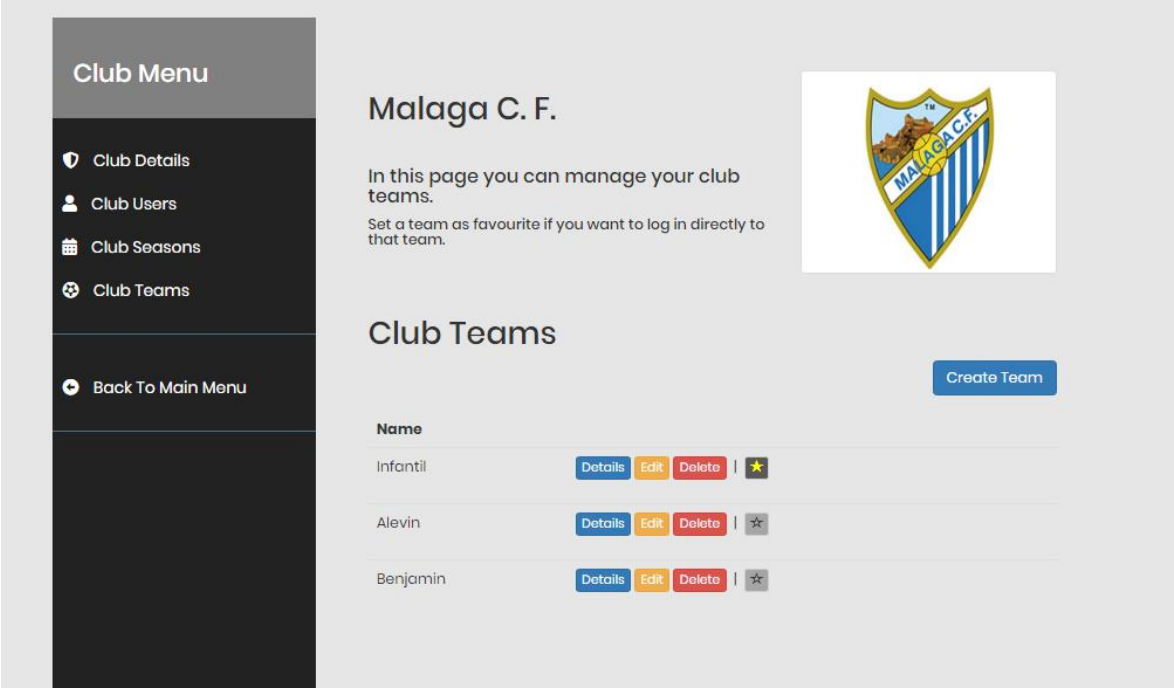


*Figura 12. Equipos de Club.*

La función del equipo es servir como contenedor para jugadores, eventos y partidos, siendo estos últimos el objetivo principal del producto. Por tanto, una vez creados los equipos procedemos de la misma forma, es decir, creando clases, relaciones, migraciones, controladores y vistas para administrar jugadores, eventos y partidos dentro del club.

Con los jugadores, distinguimos entre titulares y suplentes. Los titulares estarán más a mano para el usuario en la vista de partido para enlazarlos con los eventos que puedan ir ocurriendo.

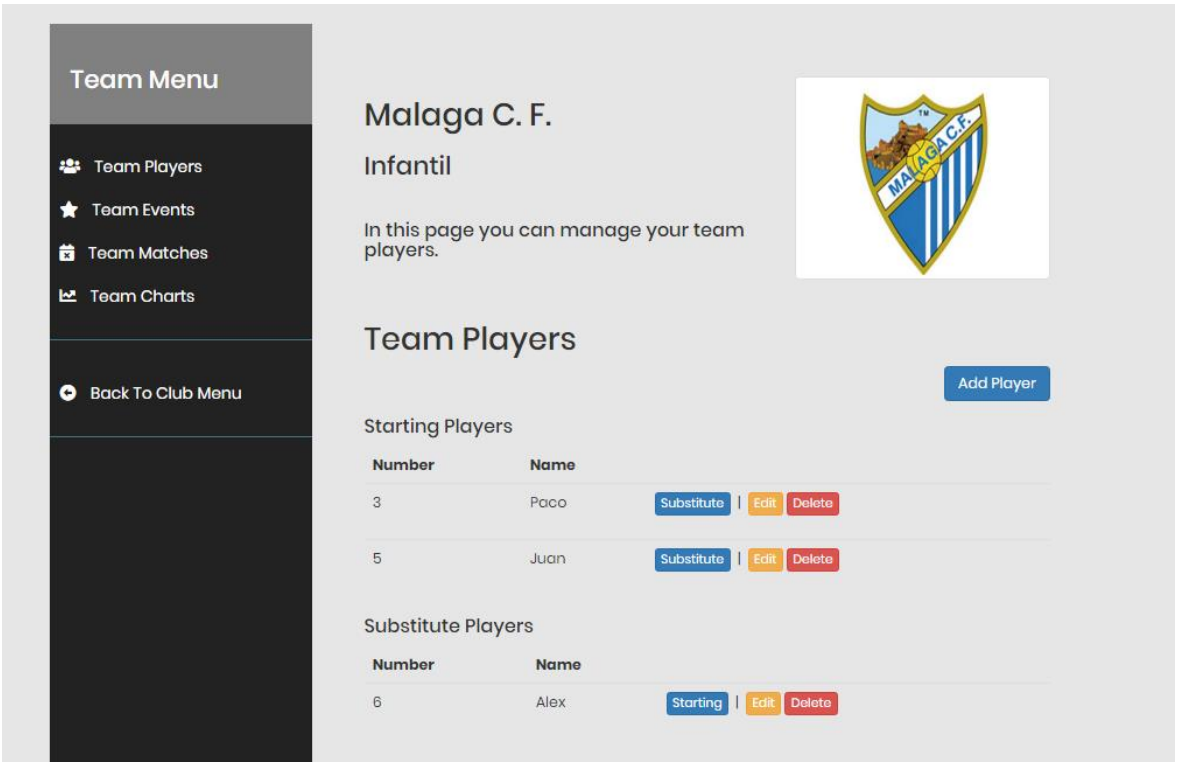


*Figura 13. Jugadores de Equipo.*

Todas estas entidades, es decir, jugadores, eventos y partidos, además de estar sujetos a un equipo de un club concreto, deberán pertenecer también a una temporada de dicho club, pues el objetivo de la aplicación es analizar el rendimiento de los jugadores a través de una temporada.

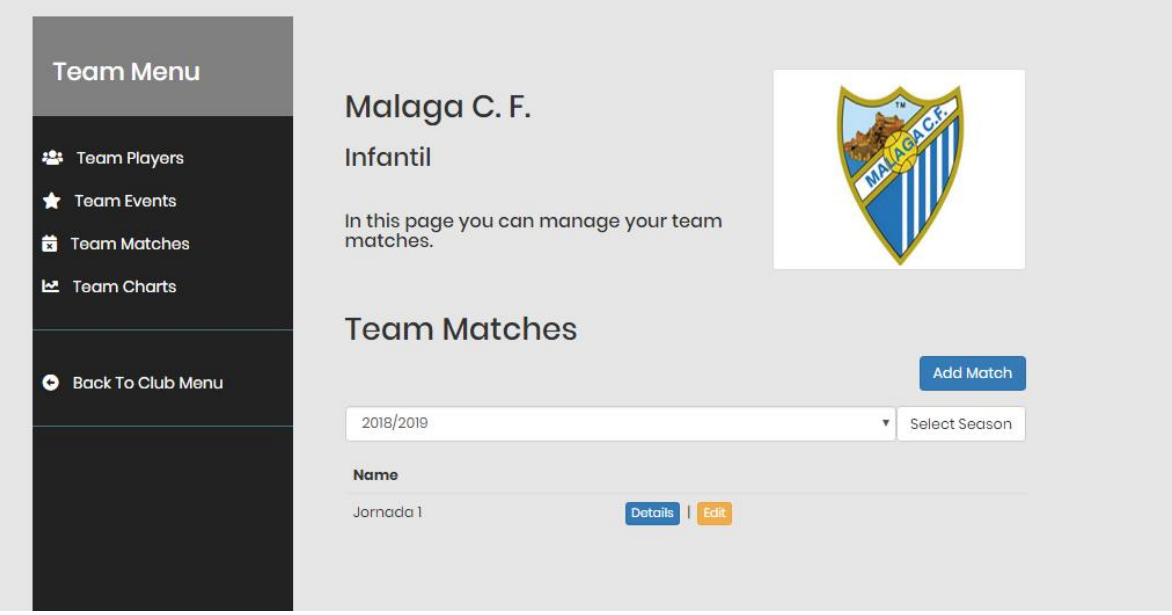


*Figura 14. Partidos de Equipo.*

Una vez añadido todo esto, obtenemos un prototipo completo donde un usuario puede registrarse en la aplicación, crear un club e invitar a otros usuarios a colaborar en el club. Tras ello, el usuario puede definir equipos dentro del club y crear, dentro de éste, jugadores, equipos y partidos.

Con esto, damos por concluida la primera iteración de la metodología ágil donde se ha obtenido una primera versión de la aplicación completamente funcional.

## 4.2 Segunda iteración

La segunda iteración es la que corresponde a la simulación en tiempo real de los partidos. Una vez dispongamos de un club con equipos, temporadas, jugadores y eventos, podremos crear partidos que contarán con un cronómetro y un marcador en tiempo real gracias al que podremos ir anotando, por cada unidad de tiempo, los eventos que ocurren y el jugador que los ejecuta.

Para hacer esto posible se han hecho uso de las tecnologías JavaScript y JQuery que permiten realizar páginas webs dinámicas, así como de la tecnología AJAX, que posibilita registrar cambios en una página sin tener que recargarla perdiendo información.

### 4.2.1 Marcador, periodos y cronómetro

Lo primero que podemos distinguir en la vista de simulación de partido es el marcador, los periodos y el cronómetro.

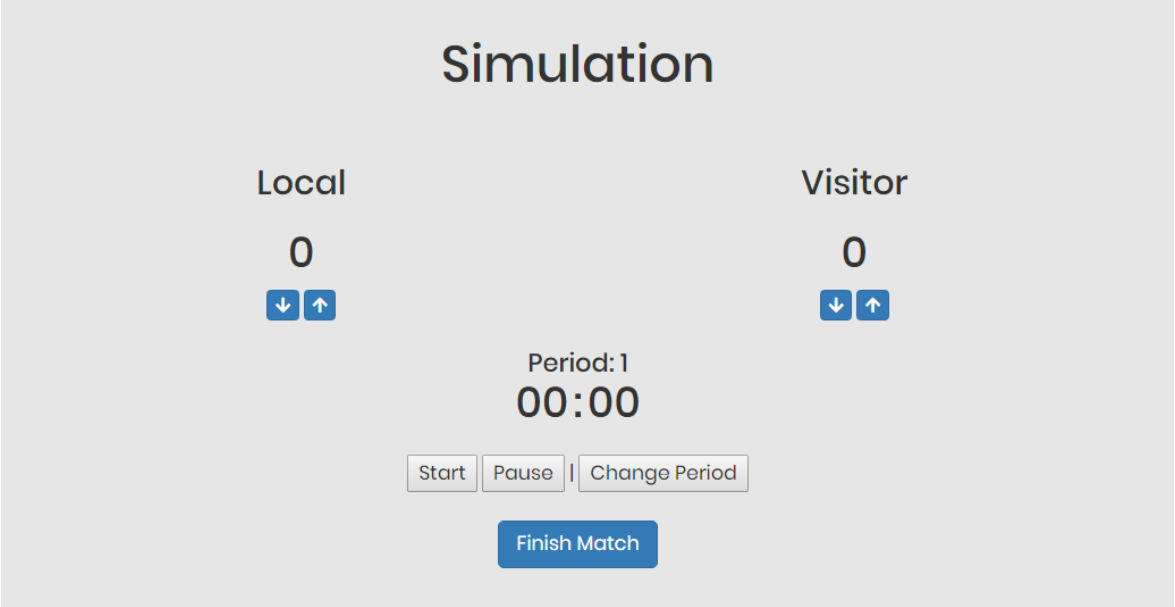


*Figura 15. Marcador de Partido.*

Como podemos ver en la Figura 15, la vista nos ofrece un marcador donde podremos ir cambiando la puntuación de los dos equipos mediante dos botones, uno para aumentarla y otro para disminuirla, para que podamos llevar un registro del resultado del partido.

A continuación, se observa un contador que indica el número de periodo o parte de partido con un botón que permite aumentarlo. Cuando el usuario pulsa el

botón de cambiar periodo, saltará un mensaje de confirmación, ya que no se podrá volver al periodo anterior.

Por último podemos ver el cronómetro, que dispone de botones para iniciarlo y pausarlo. Cuando cambiamos de periodo el cronómetro se reiniciará, volviendo a contar desde cero.

```
$(document).ready(function () {

    var myVar;

    $.ajaxSetup({ cache: false });

    window.onbeforeunload = function () {
        return 'Are you sure that you want to leave this page?';
    };

    function pad(val) { return val > 9 ? val : "0" + val; }

    $("#period").html(period);
    $("#local-score").html(local);
    $("#visitor-score").html(visitor);

    $("#startButton").click(function () {
        myVar = setInterval(function () {
            ++sec;
            $("#seconds").html(pad(sec % 60));
            $("#minutes").html(pad(parseInt(sec / 60, 10)));
        }, 1000);
        $("#startButton").attr("disabled", true);
        $('#pauseButton').removeAttr("disabled");
    });

    $("#pauseButton").click(function () {
        clearInterval(myVar);
        $("#pauseButton").attr("disabled", true);
        $('#startButton').removeAttr("disabled");
    });
```

*Figura 16. JQuery de Cronómetro.*

En la Figura 16 podemos observar que todo ello es posible gracias a funciones y variables de JavaScript y JQuery, que toman los elementos HTML y los transforman, en el caso de los contadores, o les otorgan funcionalidad, en el caso de los botones.

### 4.2.2 Jugadores y eventos

Justo debajo del marcador y el cronómetro encontramos dos tablas, una al lado de la otra. La primera tabla ofrece un listado de los jugadores titulares del partido y la segunda, un listado de los eventos que pueden ocurrir.


*Figura 17. Jugadores y Eventos.*

Si una ocurrencia de un evento tiene lugar durante el partido, el usuario puede hacer clic en el evento y a continuación en el jugador que lo haya realizado (o viceversa) para registrar la ocurrencia en el sistema.

Esto se ha realizado, una vez más, con las tecnologías JavaScript y JQuery pero con la particularidad de que se ha realizado la implementación de una máquina de estados escrita en dichos lenguajes.

```
<script>

    var state = 0;
    var playerID;
    var playerName;
    var eventID;
    var eventName;
```

*Figura 18. Variables de Estado.*

La variable state registra el estado actual del sistema. Si state está a cero, quiere decir que no hay ningún jugador ni evento seleccionado. La variable state a uno significa que se ha seleccionado un jugador. Por último, si la variable está a dos, es que se ha seleccionado un evento.

Si está a uno, es decir, si se ha seleccionado un jugador, y después se selecciona un evento, la ocurrencia se registra en el sistema y la variable vuelve al estado inicial de cero. Lo mismo ocurre si está a dos y se selecciona un jugador.

Como se puede ver en la Figura 19, cuando registramos la ocurrencia de un evento recibiremos un mensaje del sistema informándonos del suceso.

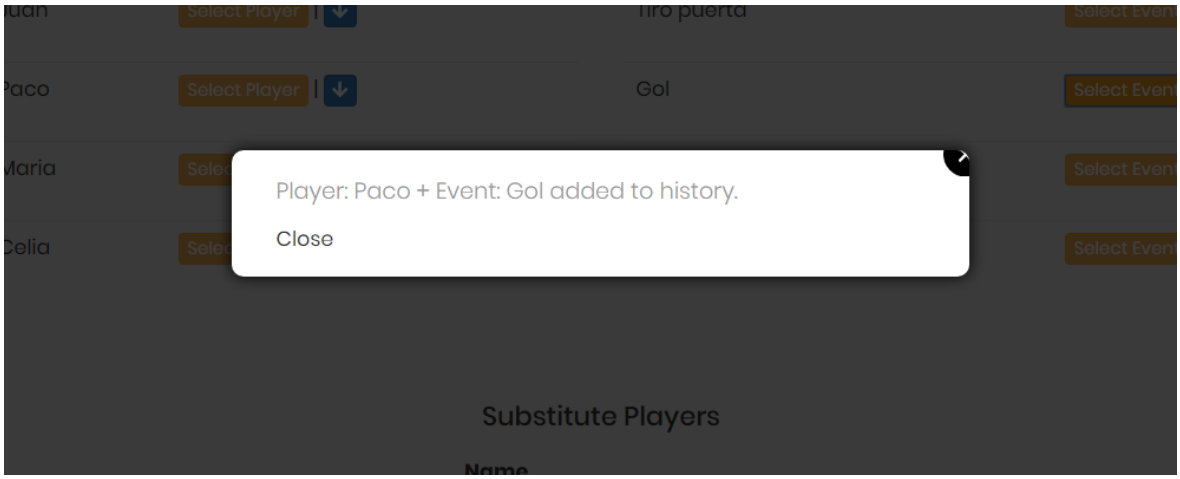


*Figura 19. Mensaje de Información*

El registro de una ocurrencia en el sistema, se realiza gracias a la tecnología AJAX. Cuando seleccionamos un jugador y un evento, se dispara una llamada a un método del controlador que realiza los cambios necesarios en la base de datos.

Las variables que se envían a dicho método se recogen de la vista por medio de JQuery. Cuando se termina de ejecutar el método, la vista se actualiza con los resultados sin necesidad de recargar la página.

```
} else if (state == 2) {
    $("#row-player_@player.ID").css("background-color", "#99ccff");

    $('#inputEvent-player_@player.ID').attr('value', eventID);
    $('#inputPeriod-player_@player.ID').attr('value', period);
    $('#inputMinute-player_@player.ID').attr('value', parseInt($("#minutes").html()));
    $("#inputSecond-player_@player.ID").attr('value', parseInt($("#seconds").html()));

    playerID =  @player.ID;
    playerName = "@player.Name";

    //llamada AJAX
      $.ajax({
        url: this.action,
        async: true,
        type: this.method,
        data: $(this).serialize(),
        cache: false,
        success: function (result) {
          //$("#substitute_@player.ID").off('submit');
          $("#divMatchEvents").html(result);
        }
    });
```

*Figura 20. Llamada AJAX.*

### 4.2.3 Sustitución de jugadores

Substitute Players

| Number | Name |
|---|---|
| 12 | Marta |
| 14 | Nicolas |
| 15 | Andres |

*Figura 21. Jugadores Suplentes.*

Más abajo en la vista de partido podemos encontrar los jugadores suplentes que cuentan con un botón para pasarlos a titulares. Asimismo, los jugadores titulares cuentan con otro botón para llevarlos a la suplencia.

Las sustituciones, como los registros de eventos, se realizan mediante una llamada AJAX al controlador que cambia el estado del jugador. Los resultados también se muestran en la vista sin recargar la página, en tiempo real.

### 4.2.4 Historial de eventos

Lo último que podemos destacar en la vista de simulación de partidos es el historial de eventos.

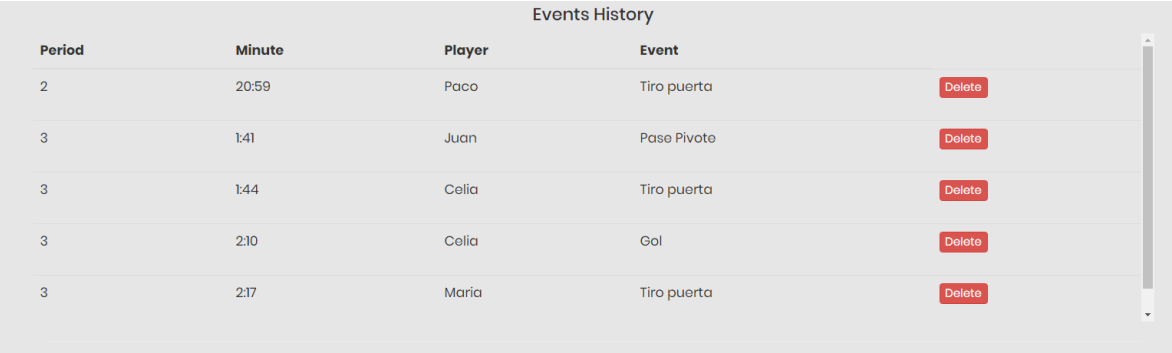


*Figura 22. Historial de Eventos.*

Cuando se registra la ocurrencia de un evento en el sistema, dicha ocurrencia aparecerá en la tabla que podemos ver en la Figura 22. La tabla se actualiza mediante AJAX cuando registramos un evento.

Cada fila cuenta con un botón de borrado por si el usuario considera oportuno eliminar la ocurrencia en caso de equivocación o similar. El borrado también realiza una llamada AJAX a un método del controlador que elimina la ocurrencia del sistema y actualiza la vista en tiempo real.

## 4.3 Tercera iteración

En la tercera se han implementado todas aquellas funcionalidades relativas al análisis estadístico. Esto incluye, principalmente, la integración del lenguaje de programación R en el sistema.

El objetivo de esto es ofrecer al usuario un análisis estadístico de la evolución de sus jugadores a lo largo de la temporada. Cada vez que se juega un partido, los datos de las ocurrencias de eventos se almacenan y se añaden a los del resto de la temporada.

Tras esto, los datos se procesarán con R, cuyo motor estará integrado con la aplicación. El motor de R generará un archivo RMarkdown, que es un lenguaje de marcado para R y que permite presentar los resultados del análisis en HTML.

Una vez procesados los datos y generados los resultados, el usuario podrá entrar, cada vez que lo desee, en el apartado de gráficos de su equipo para ver las estadísticas de la temporada.

### 4.3.1 Instalación de R.NET

R.NET es un motor de procesamiento de código R que permite la interoperabilidad entre código R y cualquier código compatible con .NET (C# en el caso que nos ocupa).

Por tanto, para poder leer y procesar código R en una aplicación .NET es necesario descargar e instalar la extensión de R.NET.

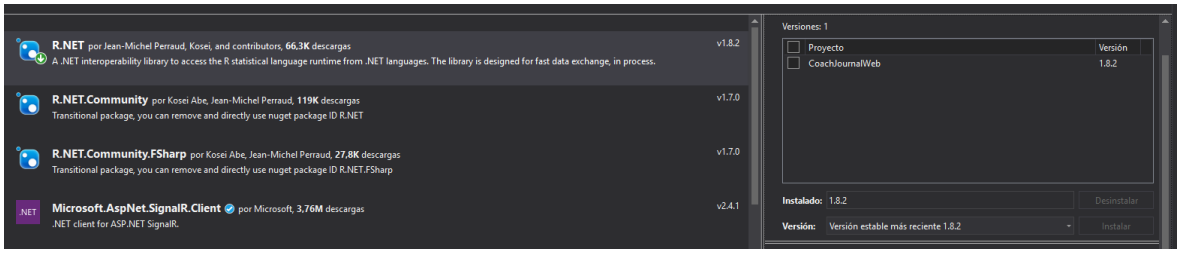


*Figura 23. Instalación de Extensiones.*

Para instalarla en nuestro proyecto, solo tenemos que ir al apartado correspondiente a las extensiones en Visual Studio, buscar la extensión que queremos e instalar.

### 4.3.2 Patrón singleton y exclusión mutua

Un vistazo rápido a la documentación nos alerta de que solo puede haber una instancia del motor R.NET corriendo en la aplicación, lo cual quiere decir que, dos usuarios no podrían utilizar al mismo tiempo el procesamiento estadístico con R.

Tal y como está diseñada la aplicación, esto plantea un problema muy serio, ya que la aplicación fue concebida para soportar múltiples usuarios al mismo tiempo.

Por tanto, la solución a esto ha sido realizada mediante dos procedimientos, la implementación del patrón singleton y la implementación de un sistema de exclusión mutua que impide a dos usuarios simultáneas acceder al motor.

En consecuencia, el primer paso ha sido envolver el uso del motor R.NET en una clase que implemente el patrón singleton, para evitar siempre que se inicialicen dos instancias del mismo, ya que esto puede causar fallos fatales en el sistema.

```
public class RNetEngine : IDisposable
{
    private bool disposed;
    private REngine engine;
    private static readonly object Lock = new object();

    private REngine Engine
    {
        get
        {
            if (engine == null)
            {
                lock (Lock)
                {
                    if (engine == null)
                    {
                        engine = REngine.GetInstance();
                    }
                }
            }
            return engine;
        }
    }
```

*Figura 24. Clase Singleton.*

Y el segundo paso, ha sido el de proteger todo el acceso y el uso de dicho motor mediante un mecanismo de exclusión mutua, tratando todas las operaciones de R.NET como sección crítica.

En la Figura 24 podemos observar la clase que implementa el patrón singleton y un ejemplo del uso de la exclusión mutua mediante la instrucción lock de C#, la cual también se usa para proteger el resto de operaciones de R.NET.

Con todo esto conseguimos, finalmente, que todos los usuarios usen la misma instancia del motor de R y también que dos usuarios no puedan acceder al mismo tiempo al motor, protegiendo sus operaciones mediante exclusión mutua.

### 4.3.3 Inicialización de motor y variables

Para poder realizar correctamente el análisis estadístico por medio de R en la aplicación, así como para poder mostrar los resultados vía HTML, primero debemos inicializar el motor, y una serie de variables y librerías.

```
private void OnStart()
{
    RNetEngine engine = new RNetEngine();
    //engine.Initialize();
    engine.Evaluate("install.packages('rmarkdown', repos = 'http://cran.us.r-project.org')");
    engine.Evaluate("install.packages('dplyr', repos = 'http://cran.us.r-project.org')");
    engine.Evaluate("library(rmarkdown)");
    engine.Evaluate("library(knitr)");
    engine.Evaluate("library(dplyr)");
    engine.Evaluate("Sys.setenv(RSTUDIO_PANDOC = 'C:/Users/Jairo/Desktop/TFG/Pandoc')");
    engine.Evaluate("Sys.setlocale('LC_ALL', 'en_US.UTF-8')");
    engine.Evaluate("Sys.setenv(LANGUAGE ='en')");
}
```

*Figura 25. Inicialización de Variables.*

La idea del análisis estadístico, como se ha comentado, es la de poder exportar y presentar los datos al usuario en HTML. Para ello, hacemos uso de RMarkdown, que es un lenguaje de marcado para código R que permite exportar la salida de los algoritmos en distintos formatos.

Esta inicialización de variables y librerías, debe hacerse al arrancar la aplicación. Por tanto, como puede verse en la Figura 25, se ha definido un método OnStart en la clase Startup de la aplicación que se ejecutará una sola vez en el arranque del sistema.

Una de las primeras cosas que debemos hacer es instalar el paquete necesario, así como sus librerías asociadas (“knitr”y “rmarkdown”). También haremos uso de una librería de manipulación de datos llamada “dplyr” de la que también instalaremos sus paquetes necesarios.

Otro elemento vital que necesitamos para generar los documentos HTML es “pandoc” que permite convertir datos de un lenguaje de marcado a otro. Por tanto, también debemos indicarle al sistema la ubicación de nuestra instalación de “pandoc” en su versión más reciente.

Por último, como podemos ver también en la Figura 25, establecemos el idioma y la codificación de caracteres.

### 4.3.4 Análisis estadístico de los partidos

Como se ha mencionado anteriormente, el análisis estadístico de los partidos es realizado por una serie de algoritmos escritos en R y desarrollados por uno de los tutores, los cuales se han adaptado a la aplicación para que puedan mostrar los resultados deseados.

Cuando se finaliza la simulación de un partido, se lanza un método del controlador de partidos que se encarga de añadir el mismo al conjunto de partidos de la temporada y que después procesa todo ese conjunto mediante los algoritmos de R.

Lo primero que debemos tener en cuenta es que tanto el formato de entrada de los datos al algoritmo como el formato de los datos de los partidos en nuestra base de datos son diferentes.

Por tanto, lo primero que debemos hacer para poder utilizar los algoritmos de R es una transformación de los datos. Podemos ver un extracto de dicha transformación en la Figura 26.

```
foreach (Event e in events)
{
    str = "c(";
    j = 0;
    foreach (Match m in matches)
    {
        List<MatchEvent> matchEvents = await matchesDb.GetMatchEvents(m);
        foreach (MatchEvent me in matchEvents)
        {
            if (j != size - 1)
            {
                if (me.Event.Name == e.Name)
                {
                    str += "1, ";
                }
                else
                {
                    str += "NA, ";
                }

            }
```

*Figura 26. Transformación de Datos.*

Una vez hecha la transformación, podemos introducir nuestros datos, es decir, la información relativa a los partidos de la temporada, en los algoritmos.

Como lo que queremos es visualizar los resultados de los análisis en forma de gráficas en nuestra aplicación web, antes lanzar directamente los algoritmos con el motor de R se hace uso del mismo para generar un documento Rmarkdown.

Rmarkdown es un lenguaje de marcado para R que permite exportar código R y su salida a otros formatos como PDF o HTML. Por tanto, se utiliza esta tecnología para crear un archivo HTML con las gráficas de salida de los algoritmos.

```
//Inicialización documento

engine.Evaluate("sink('wwwroot/html/stats_" + currentUser.Id+"_"+season.ID+"_"+team.ID+".rmd')");

engine.Evaluate("cat('---\n')");
engine.Evaluate("cat(\"title: 'Statistics'\n\")");
engine.Evaluate("cat('output: html_document\n')");
engine.Evaluate("cat('---\n')");

engine.Evaluate("cat('```{r setup, echo=FALSE}\n')");
engine.Evaluate("cat('knitr::opts_chunk$set(echo = TRUE)\n')");
engine.Evaluate("cat('```\n')");
```

*Figura 27. Inicialización del Documento.*

En la Figura 27 podemos apreciar que, mediante la instrucción sink en R podemos crear un documento Rmarkdown y, a continuación, con la instrucción cat vamos escribiendo en el documento.

El siguiente paso, en consecuencia, es el de escribir los algoritmos en el documento y proporcionarles como entrada los datos de los partidos que hemos transformado previamente.

```
engine.Evaluate("cat('```{r, include=FALSE}\n')");
bool first = true;
List<Player> players = await playersDb.GetSeasonAndTeamPlayers(season, team);
foreach (Player p in players)
{

    if (first == true)
    {
        engine.Evaluate("cat(\"datjug <- data.frame('" + p.Name + "', '" + p.Number.ToString() + "')\n\")");
        engine.Evaluate("cat(\"names(datjug) <- c('nombre', 'dorsal')\n\")");
        first = false;
    }
    else
    {
        engine.Evaluate("cat(\"da <- data.frame('" + p.Name + "', '" + p.Number.ToString() + "')\n\")");
        engine.Evaluate("cat(\"names(da) <- c('nombre', 'dorsal')\n\")");
        engine.Evaluate("cat('datjug <- rbind(datjug, da)\n')");
    }

}
engine.Evaluate("cat('names(datjug) = as.character(datjug$dorsal)\n'");
engine.Evaluate("cat('```\n')");
```

*Figura 28. Ejemplo de R.NET.*

En el ejemplo que se ve en la Figura 28, recogemos el conjunto de los jugadores de un equipo de la base de datos mediante una consulta. Tras ello, continuamos escribiendo el documento Rmarkdown y en él introducimos dicho

conjunto de jugadores en una estructura de datos propia de R para su posterior uso en el procesamiento estadístico.

Este ejemplo muestra de forma ilustrativa la interoperabilidad entre los lenguajes C# y R gracias al motor R.NET. Además, se puede apreciar cómo se va escribiendo poco a poco el documento Rmarkdown.

Una vez se termina de escribir el documento con todos los datos de entrada y los algoritmos, debemos cerrarlo mediante otra instrucción sink, tal y como puede verse en la Figura 29.

```
//Finalizamos documento

engine.Evaluate("sink()");
engine.Evaluate("rmarkdown::render('wwwroot/html/stats_" + currentUser.Id + "_" + season.ID + "_" + team.ID + ".rmd')");
```

*Figura 29. Cierre del Documento.*

Tras esto, se ejecutarán los algoritmos de R y se generará el documento HTML de salida mediante la instrucción render de Rmarkdown. Con ello, tendremos a nuestra disposición el análisis completo de la temporada en forma de gráficas y en formato HTML para presentarlo al usuario en el apartado correspondiente.

Para ver las estadísticas de la temporada, el usuario podrá irse al apartado “Team Charts” de las opciones de su equipo donde la aplicación tomará el HTML generado mediante R y Rmarkdown y lo mostrará por pantalla.

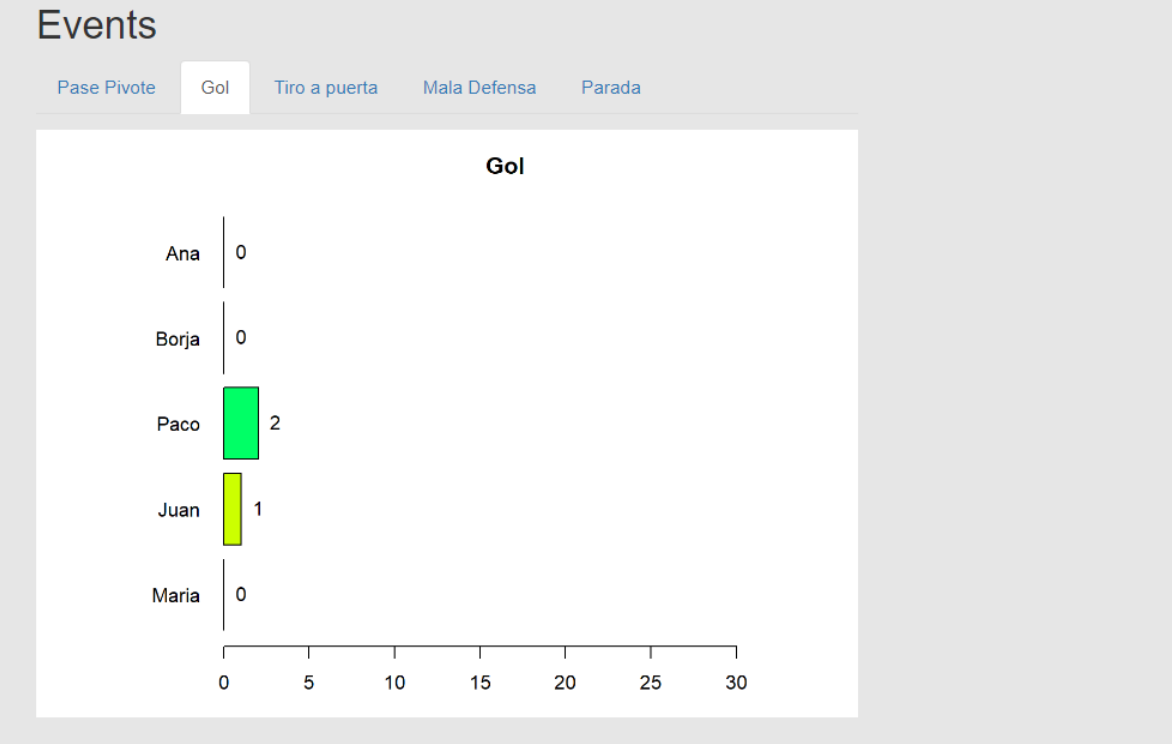


*Figura 30. Resultados de Eventos.*

La aplicación presentará, en primer lugar, los distintos eventos y las ocurrencias que ha tenido por cada jugador en la temporada, como puede verse en la Figura 30. El usuario podrá ir navegando por las distintas pestañas superiores para acceder a los distintos eventos.

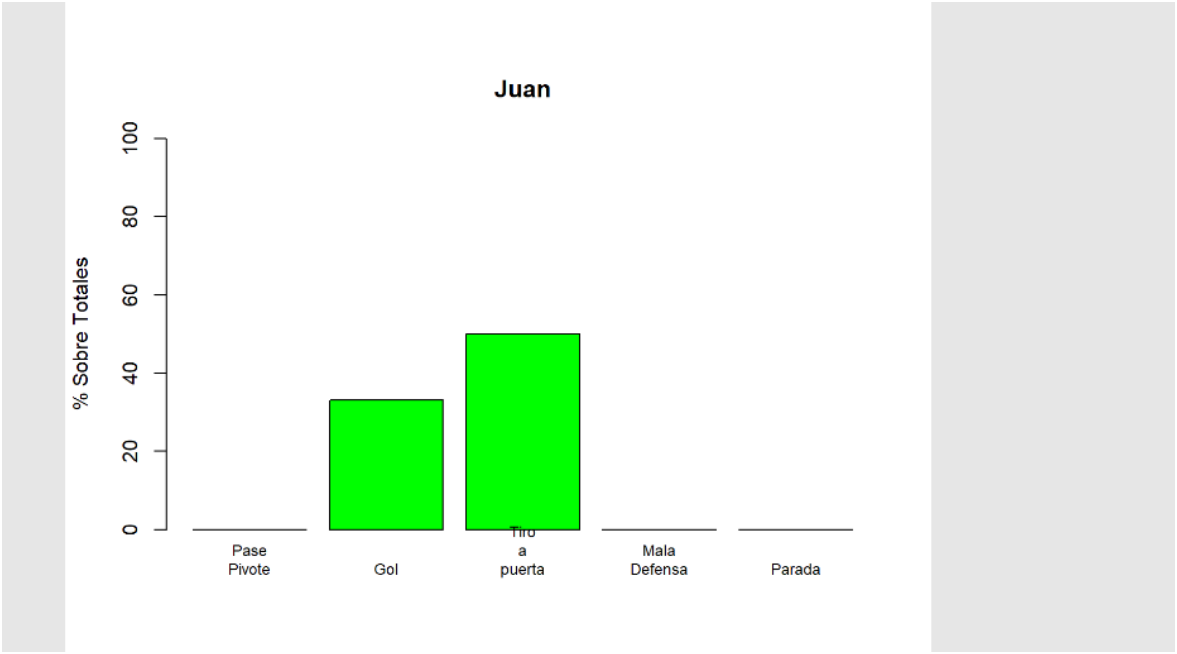


*Figura 31. Resultados de Jugador.*

En segundo lugar y como puede verse en la Figura 31, la aplicación mostrará los eventos que ha ejecutado cada jugador por medio de un porcentaje sobre el total del equipo.

Con todo esto, damos por finalizada la tercera y última iteración donde se ha hecho uso del motor R.NET para tomar datos de la aplicación, procesarlos mediante algoritmos escritos en R, generar un documento Rmarkdown y su correspondiente HTML y mostrar los resultados al usuario en forma de gráficas.

## 4.4 Otros aspectos de la implementación

Se han implementado también otras funcionalidades durante el desarrollo que han servido de ayuda para alcanzar los objetivos de la aplicación.

No se ha hablado explícitamente de ellas en los apartados anteriores por no estar directamente relacionadas con las iteraciones pero son igualmente determinantes para el éxito del producto.

A continuación se detallan algunas de ellas.

### 4.4.1 Subida de imágenes

De cara a ofrecer una sensación de personalización más profunda al usuario, se ha implementado un sistema de subida de imágenes que permite añadir un escudo o logo a un club determinado.

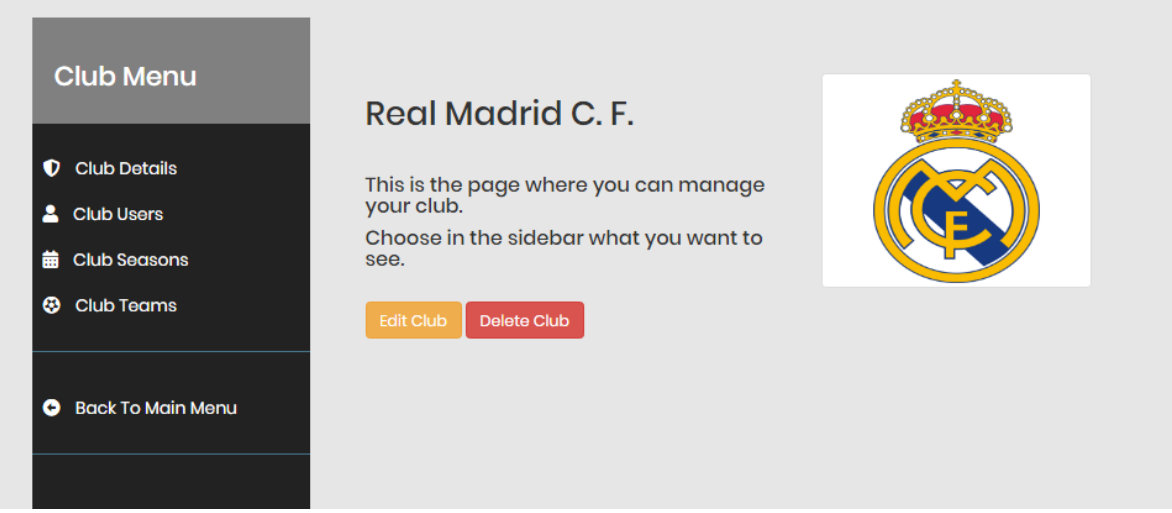


*Figura 32. Detalles de Club.*

A la hora tanto de editar como de crear un club, se ofrecerá una vista al usuario que le dará la posibilidad de elegir una imagen de su disco duro para cargarla en el servidor y asignarla al club.

Esto ha sido posible gracias a la implementación de una clase llamada ImageUploader que recoge una imagen dada y la sube al servidor en una carpeta de imágenes.

```
public class ImageUploader
{
    private IHostingEnvironment he;

    public ImageUploader (IHostingEnvironment hostingEnvironment)
    {
        he = hostingEnvironment;
    }

    public async Task<string> uploadClubImage(IFormFile file)
    {
        string filePath = "";

        if (file != null)
        {
            if (file.Length > 0)
            {
                var folder = Path.Combine(he.WebRootPath, "images/clubs");
                filePath = Path.Combine(folder, file.FileName);
                using (var fileStream = new FileStream(filePath, FileMode.Create))
                {
                    await file.CopyToAsync(fileStream);
                }
            }
        }
        return filePath;
    }
}
```

*Figura 33. Clase ImageUploader.*

Tras esto, se guarda la ruta de la imagen y su nombre en la base de datos como atributo del club para poder recuperarla y mostrarla cuando sea necesario.

El usuario podrá cambiar la imagen de su club siempre que quiera o incluso no asignar ninguna si así lo desea.

### 4.4.2 Interfaz responsiva

Uno de los requisitos principales de la aplicación es que pueda usarse tanto desde dispositivos móviles como desde ordenadores de sobremesa. Por este motivo, se ha puesto hincapié en que la interfaz se adapte a todos los tamaños de pantalla posibles.

Para ello, se ha hecho uso de las tecnologías Bootstrap y JavaScript que permiten la modificación de los distintos elementos de la interfaz para adaptarlos a los distintos tamaños de pantalla sin perder información.

Vamos a servirnos de una comparativa para poder analizar el resultado final del uso de estas tecnologías. La Figura 34 corresponde a la vista de los clubes pertenecientes a un usuario tal y como se vería en una pantalla ancha, es decir, en un ordenador de sobremesa o similar.

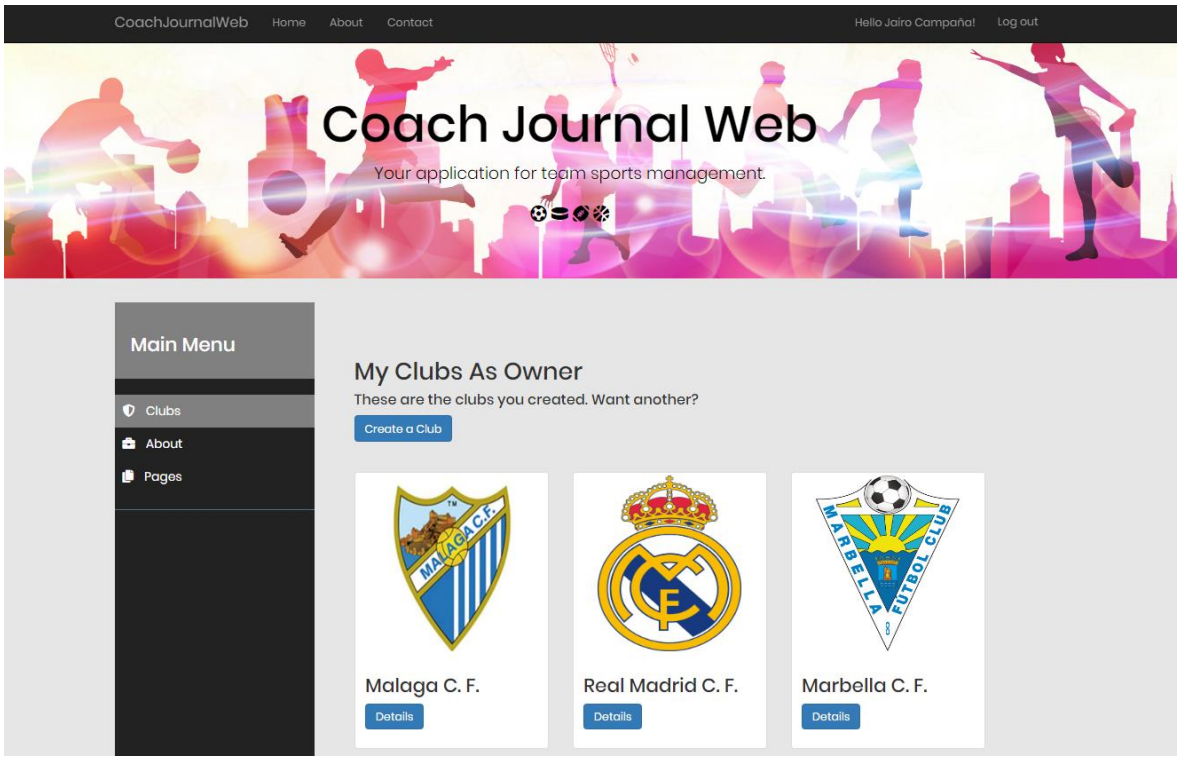


*Figura 34. Mis Clubes en Pantalla Ancha.*

Y la Figura 35 corresponde a esa misma vista, pero visto en una pantalla estrecha, por ejemplo la de un dispositivo móvil o tablet.

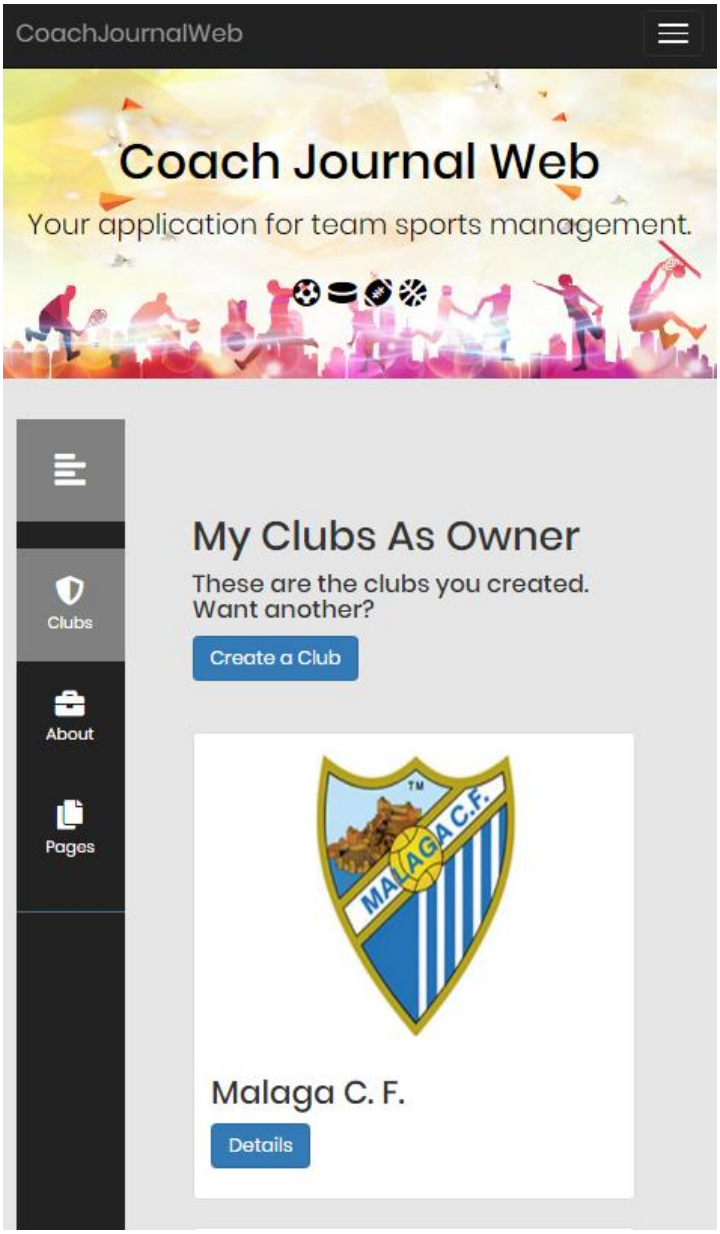


*Figura 35. Mis Clubes en Móvil.*

Como puede verse, al presentar la vista en un dispositivo móvil, esta se ha adaptado automáticamente para que pueda distinguirse igual de bien, sin pérdida de información, gracias a la implementación de las tecnologías Bootstrap y JavaScript.

Lo primero que se puede observar es que la barra de navegación superior ya no muestra los distintos enlaces, sino que se ha colapsado para no ocupar tanto

espacio y ahora los enlaces pueden desplegarse mediante un botón si el usuario lo estima oportuno.

Algo parecido ha ocurrido con la barra de navegación lateral, la cual ha encogido de forma considerable para no molestar al usuario, incluso incluyendo un botón para hacerla desaparecer por completo si así se desea.

También podemos ver que ahora los clubes no se muestran uno al lado del otro sino que dicha tabla también se ha colapsado para ofrecer una vista más compacta y ahora ofrece los clubes uno debajo del otro.

Por último, cabe destacar que también el banner que sirve como cabecera para la aplicación también se ha adaptado al tamaño de la pantalla así como todos los textos también lo han hecho.

### 4.4.3 Autorización

Con objeto de proporcionar una seguridad más fuerte a la aplicación, se ha desarrollado un sistema de autorización para todos los controladores y sus vistas asociadas.

Este sistema de autorización se encarga, en concreto, de que un usuario no pueda acceder a los datos de otro usuario. Para poner un ejemplo, sin este sistema alguien podría meterse a la aplicación y construir una URL que borrase un club que no es de su propiedad. Esto no debe permitirse bajo ningún concepto.

```
if (!User.Identity.IsAuthenticated)
{
    return RedirectToAction("Index", "Home");
}

var userId = db.httpContextAccessor.HttpContext.User.FindFirst(ClaimTypes.NameIdentifier).Value;
ApplicationUser currentUser = await usersDb.GetUserById(userId);

Team team = await teamsDb.GetTeamById(teamID);
Club club = await clubsDb.GetClubById(clubID);
Season season = await seasonsDb.GetSeasonById(seasonID);
List<Season> seasons = await seasonsDb.GetClubSeasons(club);

bool isOwner = await clubsDb.IsOwner(currentUser, club);
bool isMember = await clubsDb.IsMember(currentUser, club);

if (!isOwner && !isMember)
{
    return RedirectToAction("NotAuthorized", "Main");
}
```

*Figura 36. Código de Autorización.*

Como puede verse en la Figura 36, se ha implementado un código de autorización que, cada vez que se ejecuta un método en cualquier controlador, comprueba si la persona que accede se ha autenticado en la aplicación.

Tras esto, traemos de la base de datos el club al que quiere acceder y comprobamos si el usuario es miembro o es dueño, dicho de otro modo, si tiene acceso al club.

Si lo tiene, se le deja continuar pero si no lo tiene se le llevará a una vista que le indicará que no está autorizado a acceder a esos datos y no se le dejará seguir con la operación.

### 4.4.4 Navegación

Para facilitar la navegación por la aplicación web, se ha visto conveniente implementar una serie de barras de navegación que se podrán encontrar en la mayoría de vistas.

Con ello, se ha buscado que el usuario tenga siempre a mano las acciones más importantes para cada momento, evitándole así el tedio de tener que ir hacia atrás o hacia delante en el navegador para cada operación que quiera realizar.


*Figura 37. Barra de Navegación Superior.*

En la Figura 37, se puede observar la barra superior, donde podremos realizar acciones como ir a la página principal de la aplicación, acceder a los datos de nuestra cuenta o cerrar nuestra sesión.

También se han implementado barras laterales que ayuden al usuario con el resto de las acciones. La Figura 38 muestra la barra lateral de la página de inicio de la aplicación.

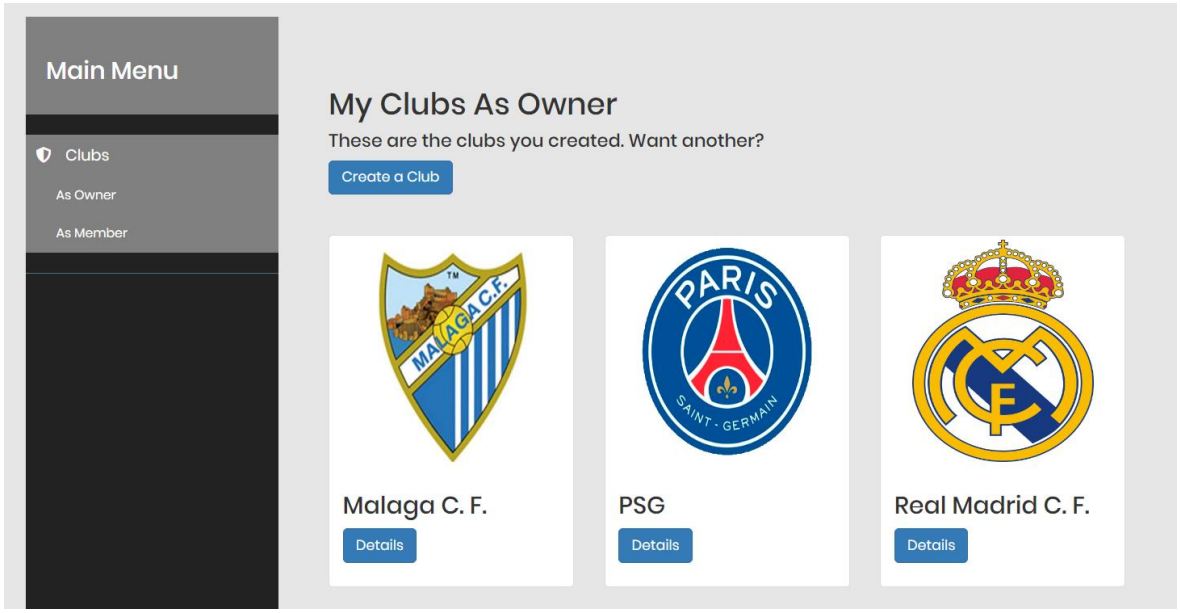


*Figura 38. Barra Lateral Inicial.*

Con ella podremos acceder a nuestros clubes, los que tengamos registrados como propietarios o los que tengamos registrados como miembros.

Si accedemos a alguno de nuestros clubes, entraremos en el siguiente nivel y se nos presentará la siguiente barra de navegación, que podemos ver en la Figura 39.


*Figura 39. Barra Lateral de Club.*

Como ya estamos en el menú de club, la barra nos permitirá realizar operaciones relativas a este, así como volver atrás al menú anterior.

Podremos ver los detalles del club, donde también podremos editarlo y borrarlo, y podremos invitar a otros usuarios a colaborar con el club como administradores o como miembros. Podremos también acceder a las temporadas del club y definir otras nuevas o acceder a los equipos del club donde se nos permitirá crearlos, editarlos y borrarlos.

Si seleccionamos un equipo de un club entraremos en el siguiente nivel y veremos una nueva barra de navegación lateral, esta vez relativa a las operaciones que se pueden realizar para un equipo. Podemos verla en la Figura 40.

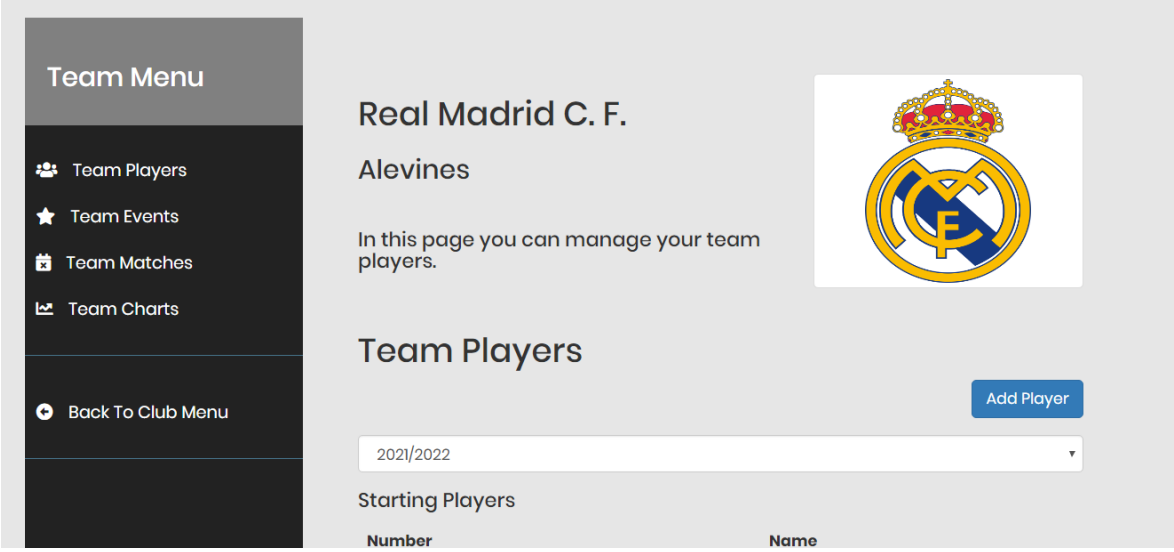


*Figura 40. Barra Lateral de Equipo.*

La barra nos permitirá realizar acciones como acceder a los jugadores, eventos, partidos y estadísticas de un equipo.

Cabe destacar, como se ha visto en uno de los apartados anteriores, que las barras de navegación implementadas son totalmente responsivas y se adaptan a distintos tamaños de pantalla gracias a la tecnología JavaScript.

### 4.4.5 Equipo favorito

Si un usuario es miembro de un club, es decir, no es administrador y solo entra en la aplicación para gestionar un solo equipo, para acceder a este tendría que navegar desde la página de inicio, elegir los clubes de los que es miembro, entrar en un club, entrar en los equipos de ese club y seleccionar su equipo.

Tras un tiempo de pruebas, se comprobó que esto era demasiado tedioso para un usuario que accede frecuentemente a la aplicación y no quiere preocuparse de los demás aspectos del club, es decir, si solo quiere preocuparse de su equipo.

Por tanto, se ideó un sistema para evitar todo esto al usuario mediante un sistema de equipo favorito.

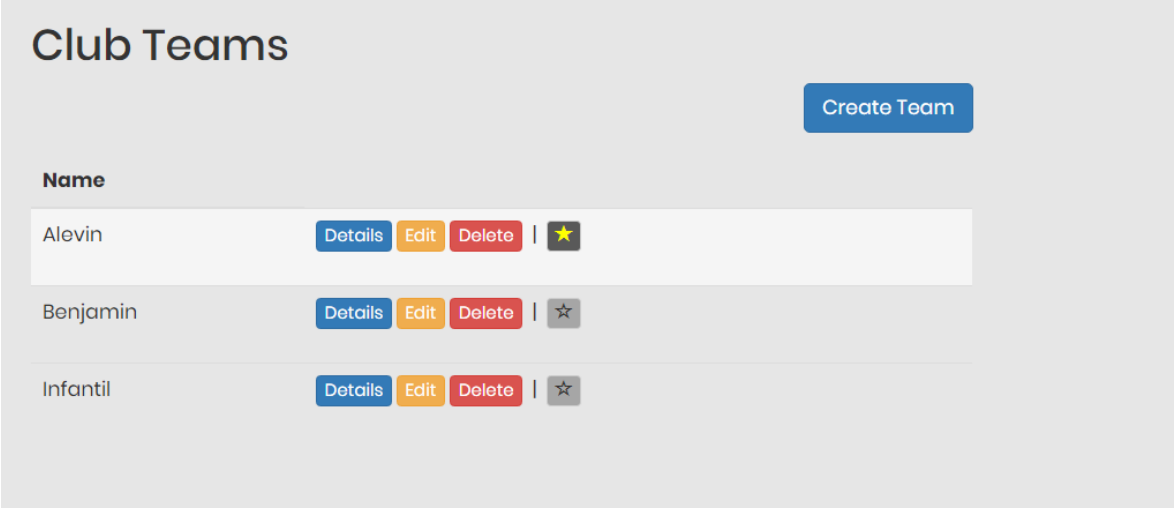


*Figura 41. Equipo Favorito.*

Si un usuario elige un equipo como favorito, la aplicación directamente le redirigirá a la vista de dicho equipo cuando inicie sesión, evitándole así el tedio de tener que navegar hasta él cada vez.

El usuario también podrá cambiar de equipo favorito a su gusto e incluso no tener ningún equipo favorito seleccionado.

### 4.4.6 Acceso a la base de datos

En aplicaciones ASP.NET, la forma normal de acceder a la base de datos sería mediante una llamada a esta desde cualquier controlador de la aplicación.

El problema que presenta este sistema, es que el código está totalmente acoplado, es decir, estamos mezclando la lógica de la aplicación o de negocio con la capa de accedo a la base de datos.

Esto tiene el inconveniente de que, por ejemplo, si queremos realizar algún cambio en el acceso a la base de datos, como implementar consultas nuevas, debemos modificar nuestro código en cada una de las operaciones y en cada uno de los controladores donde hayamos escrito dicho acceso.

Por este motivo, se ha optado por realizar una capa de acceso de datos para cada entidad del sistema y utilizar dicha capa para realizar o implementar las consultas a la base de datos, reduciendo así el acoplamiento de código..

```
public class EventsDbAccess
{
    private ApplicationDbContext db;

    public EventsDbAccess (ApplicationDbContext applicationDbContext)
    {
        db = applicationDbContext;
    }

    public async Task<Event> GetEventById(int? eventID)
    {
        Event ev = await db.Event.SingleOrDefaultAsync<Event>(e => e.ID == eventID);
        db.Entry<Event>(ev).Reference(e => e.Season).Load();
        return ev;
    }

    public async Task<bool> CreateEvent (Event ev, Team team, Season season)
    {
        ev.Team = team;
        ev.Season = season;

        //Commit changes to database
        db.Add(ev);
        await db.SaveChangesAsync();

        return true;
    }
```

*Figura 42. Acceso a la Base de Datos.*

La forma de hacerlo ha sido implementando una clase de acceso a la base de datos por cada entidad. Podemos ver un ejemplo para la entidad evento en la Figura 42.

En esta clase se engloban todos los métodos correspondientes a consultas en la base de datos para una entidad. Si es necesario acceder a ella desde un controlador, traeremos una instancia de dicha clase al mismo para poder utilizar sus métodos, logrando así un código más eficiente y reutilizable.

# 5

# Pruebas

Las pruebas de software corresponden a una serie de investigaciones empíricas y técnicas cuyo objetivo es proporcionar información sobre la calidad del mismo, así como advertir a las partes interesadas del proyecto de los posibles riesgos o fallos.

En este capítulo, por tanto, se hablará sobre el proceso de pruebas que ha seguido el proyecto que nos ocupa.

## 5.1 Consideraciones previas

Debido a la falta del tiempo necesario, no se han realizado paquetes de pruebas unitarias y automáticas para testar cada funcionalidad del sistema.

Sin embargo, como la finalidad del proyecto es proveer de una plataforma a los usuarios para que puedan gestionar sus clubes deportivos y analizar el rendimiento de sus jugadores, sí que se ha considerado oportuno realizar un test experimental a distintos usuarios que han hecho las veces de sujetos de pruebas.

A falta de pruebas automáticas, este test experimental intenta que los usuarios o, en este caso, sujetos de pruebas, comprueben si todas las

funcionalidades se comportan de la forma esperada, si el sistema responde de forma satisfactoria y, en general, si se han cumplido los objetivos de la aplicación.

## 5.2 Test experimental

Para el diseño de este test experimental, se han tomado como referencia los cuestionarios de evaluación heurística de interfaces de usuario pero se han alterado para enfocarse, a parte de en la usabilidad, en la funcionalidad y en el rendimiento del sistema.

A cada pregunta del cuestionario, el usuario podrá responder si está en desacuerdo, a lo que se asignará una puntuación de valor cero, si no está ni de acuerdo ni en desacuerdo, con puntuación de valor cinco, o si está completamente de acuerdo, con puntuación de valor diez.

A continuación se detalla el contenido del cuestionario:

| Pregunta | Puntuación |
|---|---|
| 1. A primera vista, la interfaz le ha parecido agradable. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 2. El proceso de registro le ha parecido sencillo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 3. Le ha parecido intuitivo acceder a la información de su cuenta. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 4. Le ha parecido sencillo crear un club. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 5. Le ha parecido intuitivo acceder a los detalles de su club. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo. |

| | |
|---|---|
| | 10 – Completamente de acuerdo. |
| 6. Le ha parecido fácil invitar a otros usuarios al club. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 7. Le ha parecido sencillo crear temporadas para su club. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 8. Le ha parecido sencillo crear equipos para su club. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 9. La función de equipo favorito le ha parecido útil. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 10. Le ha parecido intuitivo acceder a los detalles de su equipo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 11. Le ha parecido fácil añadir jugadores al equipo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 12. Le ha parecido fácil añadir eventos al equipo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 13. Le ha parecido fácil añadir partidos al equipo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 14. Ha encontrado fácilmente la opción de simular un partido. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 15. La interfaz de simulación de partido le parece intuitiva. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |

| | |
|---|---|
| 16. Ha sabido anotar la ocurrencia de un evento durante el partido. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 17. Ha sabido sustituir a un jugador durante el partido. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 18. Ha sabido manejar el marcador, el cronómetro y los periodos durante el partido. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 19. Le ha parecido sencillo observar las estadísticas de su equipo. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 20. En general, cree que es sencillo manejarse por la aplicación. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 21. En general, cree que la aplicación responde de forma satisfactoria a todas las acciones. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |
| 22. En general, cree que la aplicación cumple los objetivos propuestos. | 0 – En desacuerdo.<br>5 – Ni de acuerdo ni en desacuerdo.<br>10 – Completamente de acuerdo. |

*Tabla 1. Preguntas del Test.*

Como puede verse, las preguntas del test han sido diseñadas para intentar abarcar todas las acciones del sistema de forma concreta así como para medir el grado de satisfacción general con el uso de la aplicación.

## 5.3 Participantes seleccionados

Uno de los objetivos principales de la aplicación es que pueda ser usada por todo tipo de usuarios. Por tanto, de cara a obtener unos resultados cercanos a la realidad, se han escogido usuarios de distintos perfiles para probar la aplicación. Los perfiles son los siguientes:

- Usuario inexperto. Este usuario no tiene ningún tipo de conocimiento técnico pero utiliza páginas webs y aplicaciones móviles de forma esporádica. Lo llamaremos U1.

- Usuario medio. Este usuario tiene algún conocimiento técnico y es habitual que se maneje con páginas webs y aplicaciones móviles. Lo llamaremos U2.

- Usuario experto. Este usuario posee conocimientos técnicos en informática y conoce a la perfección las aplicaciones webs y móviles. Lo llamaremos U3.

Como se ha expuesto, el objetivo de este test no es solo medir la usabilidad sino también la funcionalidad y el rendimiento. Por tanto, antes de rellenar el test, se ha pedido a estos usuarios que prueben la aplicación de forma continuada durante un tiempo determinado y se les ha instado a probar todas y cada una de las funcionalidades.

Esto ha llevado a encontrar una serie de errores menores en el funcionamiento de la aplicación que se han ido solventando con el tiempo. Por ello, la labor de los participantes en las pruebas de la aplicación ha sido vital para el correcto funcionamiento de la misma.

Finalmente y tras un tiempo de pruebas, se ha pedido a los usuarios que rellenen el test para medir el grado de usabilidad, funcionalidad y rendimiento de la aplicación.

## 5.4 Resultados del test

A continuación se presentan los resultados del test por medio de una tabla donde se muestra, por cada usuario, la puntuación que le ha dado a cada pregunta así como la puntuación media que ha obtenido cada una.

| Pregunta | U1 | U2 | U3 | Puntuación Media |
|---|---|---|---|---|
| 1 | 10 | 10 | 10 | **10** |
| 2 | 10 | 10 | 10 | **10** |
| 3 | 10 | 10 | 10 | **10** |
| 4 | 10 | 10 | 10 | **10** |
| 5 | 10 | 10 | 10 | **10** |
| 6 | 5 | 10 | 10 | **8.3** |
| 7 | 10 | 10 | 10 | **10** |
| 8 | 0 | 5 | 10 | **5** |
| 9 | 10 | 10 | 10 | **10** |
| 10 | 10 | 10 | 10 | **10** |
| 11 | 10 | 10 | 10 | **10** |
| 12 | 10 | 10 | 10 | **10** |
| 13 | 10 | 5 | 10 | **8.3** |
| 14 | 5 | 5 | 10 | **6.7** |
| 15 | 10 | 10 | 10 | **10** |
| 16 | 5 | 10 | 10 | **8.3** |
| 17 | 5 | 10 | 10 | **8.3** |
| 18 | 5 | 10 | 10 | **8.3** |
| 19 | 5 | 5 | 10 | **6.7** |
| 20 | 5 | 10 | 10 | **8.3** |
| 21 | 10 | 10 | 10 | **10** |
| 22 | 10 | 10 | 10 | **10** |
| | | | Total: | **198.2** |

*Tabla 2. Resultados del Test.*

## 5.5 Conclusiones

Una vez completados los test podemos comprobar en un vistazo rápido que el grado de satisfacción ha sido mayormente positivo.

Lo primero que llama la atención es la diferencia de puntuación entre los usuarios inexperto y experto. Con esto se demuestra que es importante tener en cuenta a todo el público potencial de la aplicación, que no tiene por qué tener conocimientos informáticos.

Si entramos a valorar aspectos concretos, podemos ver que las puntuaciones más bajas son relativas a la creación de equipos, a la posición de la acción de simular partido, que los usuarios han encontrado algo difícil de localizar, y a la observación de las estadísticas del equipo. Todos ellos son aspectos que se valorarán mejorar en el futuro.

Si miramos a las preguntas que miden los aspectos más generales como el hecho de si la aplicación es intuitiva, si responde de forma satisfactoria a las acciones y si la aplicación cumple los objetivos, encontramos una puntuación bastante alta, con lo cual puede suponerse el éxito del proyecto.

Si valoramos el heurístico utilizado, vemos que la puntuación total obtenida es 198.2 de un total de 220, lo cual supone algo más del noventa por ciento de la puntuación total. Igualmente, esto es un indicativo de que el proyecto, si bien no es perfecto, se ha llevado a cabo de forma satisfactoria.

# 6

# Conclusiones

Finalizamos la memoria con este último capítulo donde analizamos, una vez terminado el proceso de diseño y desarrollo de la aplicación, los resultados del proyecto y las conclusiones que podemos obtener de ellos.

Se pondrá especial atención, en primer lugar, al hecho de si se han cumplido los objetivos últimos del proyecto. En segundo lugar, se señalarán las dificultades encontradas a lo largo del proceso de desarrollo. Y, por último, se detallarán las posibles líneas futuras que el proyecto podría seguir.

## 6.1 Objetivos cumplidos

En primer lugar, se van a analizar los objetivos cumplidos del trabajo desde el punto de vista del proyecto software.

El objetivo de la aplicación “Coach Journal”, desde su concepción, fue el de construir una plataforma para que cualquier entrenador de un equipo deportivo pudiera analizar el rendimiento de sus jugadores a lo largo de una temporada y planificar los entrenamientos en base a las necesidades de cada uno.

Tras la toma formal de requisitos, se estableció que la aplicación debía permitir al usuario crear un club, invitar usuarios a ese club, definir equipos dentro del club y añadir jugadores, eventos y partidos en el equipo. Una vez hecho esto, el usuario debía poder simular partidos y anotar las ocurrencias de eventos en tiempo real. Las ocurrencias debían poder analizarse estadísticamente y presentar una serie de gráficos al usuario donde este podría ver el rendimiento de sus jugadores a lo largo de la temporada.

Desde el punto de vista técnico, el proyecto debía realizarse en forma de aplicación web escrita en C# mediante la tecnología ASP.NET. Para el análisis estadístico, la aplicación debía integrar el lenguaje R y ser capaz de ejecutar código R con las variables recogidas de la aplicación.

En vista de todo esto y de los resultados obtenidos, se puede verificar que la aplicación ha sido desarrollada con éxito, atendiendo a los requisitos establecidos y dentro del plazo determinado.

Desde el punto de vista puramente académico se establece que el trabajo de fin de grado corresponde a la elaboración de un trabajo personal por parte del alumno donde aplique e integre sus conocimientos teóricos y técnicos con el objeto de resolver un problema de índole técnica o científica.

Atendiendo a este criterio, se puede afirmar que se han cumplido las especificaciones de lo que supone este trabajo. El alumno ha sido capaz de aplicar los conocimientos aprendidos durante el estudio de la titulación al proceso de análisis del problema, diseño de la solución y desarrollo del proyecto, aplicando conceptos teóricos y prácticos conocidos de la informática.

Por todo esto, finalmente se concluye que se han cumplido los objetivos del proyecto y del trabajo de fin de grado.

## 6.2 Dificultades encontradas

La dificultad más problemática del proyecto ha sido la integración de los algoritmos de R proporcionados por el tutor con la aplicación.

En primer lugar, tenemos que el motor R.NET no puede ser accedido al mismo tiempo por varios usuarios y, además, solo puede haber una instancia del mismo ejecutándose en el proyecto.

Para solventar esto, tal y como se ha detallado anteriormente, se ha envuelto el motor en una clase que implementa el patrón singleton, con lo que evitamos que pueda haber más de una instancia del motor en ejecución en cualquier momento. También se han protegido sus métodos mediante un mecanismo de exclusión mutua para que dos usuarios no puedan acceder al mismo tiempo a sus operaciones.

En segundo lugar, tenemos que para generar las gráficas y mostrarlas al usuario en HTML no bastaba con ejecutar código R de forma normal con R.NET, ya que la salida no podía ser recogida fuera del ámbito del motor.

Como solución, por medio del motor R.NET y cada vez que se quiere realizar un análisis de la temporada, se genera un documento Rmarkdown cuya salida podemos exportar al servidor en forma de documento HTML, el cual sí que puede presentarse al usuario.

Por último, el formato de los datos de los algoritmos proporcionados y de la aplicación no eran iguales, es decir, no eran compatibles. Por tanto, se ha tenido que realizar un proceso de conversión previo de todos los datos para adaptarlos al formato de entrada de los algoritmos.

Con todo esto se han resuelto los problemas satisfactoriamente para cumplir los objetivos del proyecto.

## 6.3 Líneas futuras

El proyecto nació con la intención de ser ampliado en futuros trabajos de fin de grado, considerándose el que nos ocupa como los cimientos del mismo.

La ampliación más evidente que puede hacerse a la aplicación es la de expandir el número de estadísticas mostradas mediante la adición de nuevos

algoritmos de R así como la modificación del código de la aplicación para adaptarse a ellos.

La más sencilla, sería la de hacer las modificaciones necesarias para obtener gráficos correspondientes a una sola jornada, en lugar de la temporada completa.

Otra ampliación que podría hacerse sería la de profundizar en los aspectos de gestión de club añadiendo más opciones o convirtiendo la página del club en una especie de red social para sus miembros, con el añadido de mensajes y otros elementos relacionados.

También podrían modificarse los partidos para que tuviesen en cuenta las convocatorias de jugadores o también qué jugador está jugando en cada momento con objeto de que ello tenga incidencia en las estadísticas, ya que, en el estado actual de la aplicación y desde el punto de vista estadístico, se toma que todos los jugadores del equipo van convocados en todos los partidos y juegan todos los minutos.

Por último se podría añadir una línea que desarrollase la funcionalidad de los partidos en forma de aplicación móvil, tal y como se había pensado para este proyecto en un primer momento.

Esto supondría un mejor rendimiento y una mejor usabilidad de las funcionalidades en dispositivos móviles y permitiría simular partidos sin necesidad constante de conexión a internet.

La aplicación móvil, tras un partido, podría enviar los resultados a la aplicación web para que el usuario pudiera visualizar los resultados en su ordenador de escritorio. De esta forma mantendríamos separados en dos aplicaciones los aspectos de simulación de partidos y los de gestión de club.

# Referencias


- Anderson, Rick (s.f.), *Documentación de ASP.NET*, Recuperado de https://docs.microsoft.com/es-es/aspnet/

- JS Foundation (s.f.), *jQuery API Documentation*, Recuperado de https://api.jquery.com/

- Otto, Mark, Thornton, Jacob and Bootstrap contributors (s.f.), *Bootstrap · The world's most popular mobile-first and responsive front-end framework.*, Recuperado de https://getbootstrap.com/docs/3.3/

- *R Documentation and manuals*, (s.f), Recuperado de https://www.rdocumentation.org/

- *R Markdown*, (s.f.), Recuperado de https://rmarkdown.rstudio.com/

- *R.NET documentation*, (s.f.), Recuperado de http://jmp75.github.io/rdotnet

- *W3Schools Online Web Tutorials*, (s.f.), Recuperado de https://www.w3schools.com/

# Apéndice

# Manual de Usuario

## Introducción:

Coach Journal es una aplicación web que permite a los usuarios gestionar equipos deportivos con el objetivo de analizar el rendimiento de sus jugadores a lo largo de una temporada.

Con ella podrá registrarse, crear un club y añadir a otros usuarios de la aplicación a que colaboren con usted. Tras esto, podrá definir equipos en el club e incluir jugadores, eventos y partidos.

Con ello podrá simular partidos en tiempo real y anotar, por cada unidad de tiempo, la ocurrencia de un evento y el jugador que lo ejecuta. Los datos del partido se analizarán por medio de algoritmos estadísticos para presentarle resultados gráficos que le posibilitarán analizar el rendimiento y la evolución de sus jugadores y, en base a ello, planificar los entrenamientos.

Con este manual se pretende informar al usuario de todas las funcionalidades del sistema y guiarlo a través de ellas para que pueda sacar el máximo rendimiento a la aplicación.

## 1. Inicio de sesión y registro de usuario

La primera vez que accedemos al sistema, lo que se nos muestra es la pantalla de inicio de sesión.


*Figura 43. Inicio de Sesión.*

Desde aquí podremos iniciar sesión si ya estamos registrados en el sistema o bien podemos registrarnos si no lo estamos.

Para poder usar la aplicación será necesario registrarse previamente. Para ello, debemos introducir nuestro correo, nuestro nombre de usuario y nuestra contraseña en la pantalla de registro.

Si queremos cerrar sesión, en todo momento tendremos la opción disponible en la barra de navegación superior.

## 2. Datos de la cuenta

Si queremos acceder a los datos de nuestra cuenta para modificarlos, solo tenemos que seleccionar nuestro nombre en la barra de navegación superior.


*Figura 44. Botón Detalles.*

Con ello la aplicación nos enviará a la vista de los detalles de la cuenta.

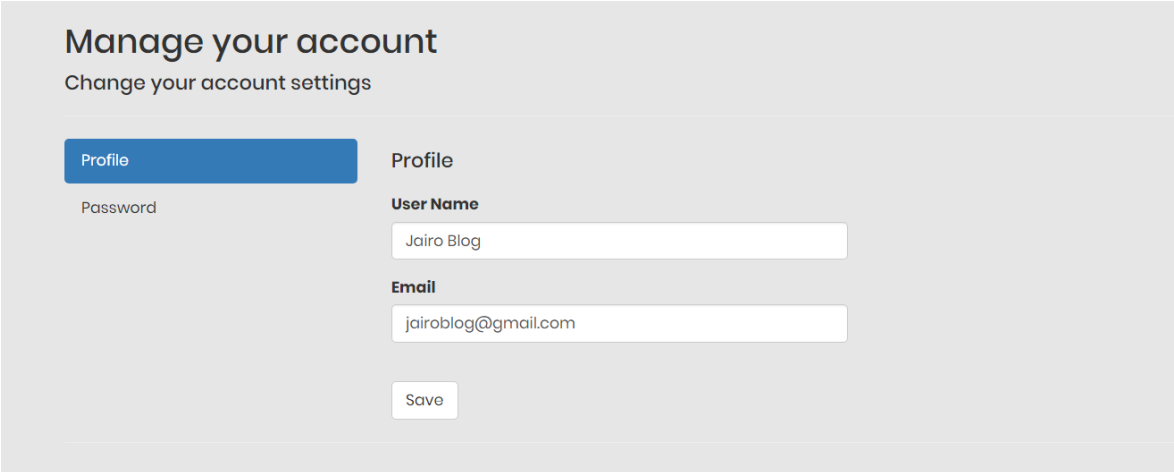


*Figura 45. Detalles de Cuenta.*

En esta vista podremos modificar cualquier aspecto de nuestra cuenta como el nombre, el correo o la contraseña.

## 3. Mis clubes

Cuando iniciamos sesión en la aplicación, se nos lleva a la vista de nuestros clubes. En concreto, se nos lleva a la vista de clubes que hayamos creado nosotros como dueños.

En Coach Journal, podremos colaborar en clubs como dueños (administradores) o como invitados en un club que haya creado otra persona. Los administradores podrán acceder a más funciones dentro del club que los invitados. Podremos entrar a nuestros clubes como dueños y como miembros desde la barra de navegación lateral.

Si nadie nos ha invitado a ser miembro de un club, no nos aparecerá ningún club en el apartado correspondiente. Por otra parte, si no hemos creado nosotros ningún club o no hemos sido invitados como administradores a otro club, tampoco nos aparecerá ningún club en el apartado de “My Clubs As Owner”.

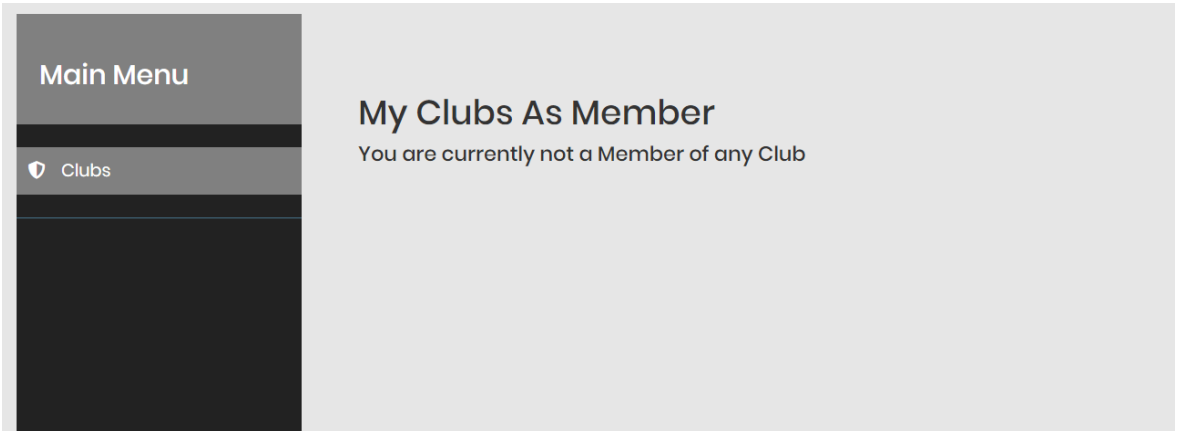


*Figura 46. Ningún Club Como Miembro.*

Si por el contrario, alguien nos ha invitado o hemos creado un club, nos apareceran los clubes en los distintos apartados y podremos acceder a ellos pulsando en el boton “Details”.

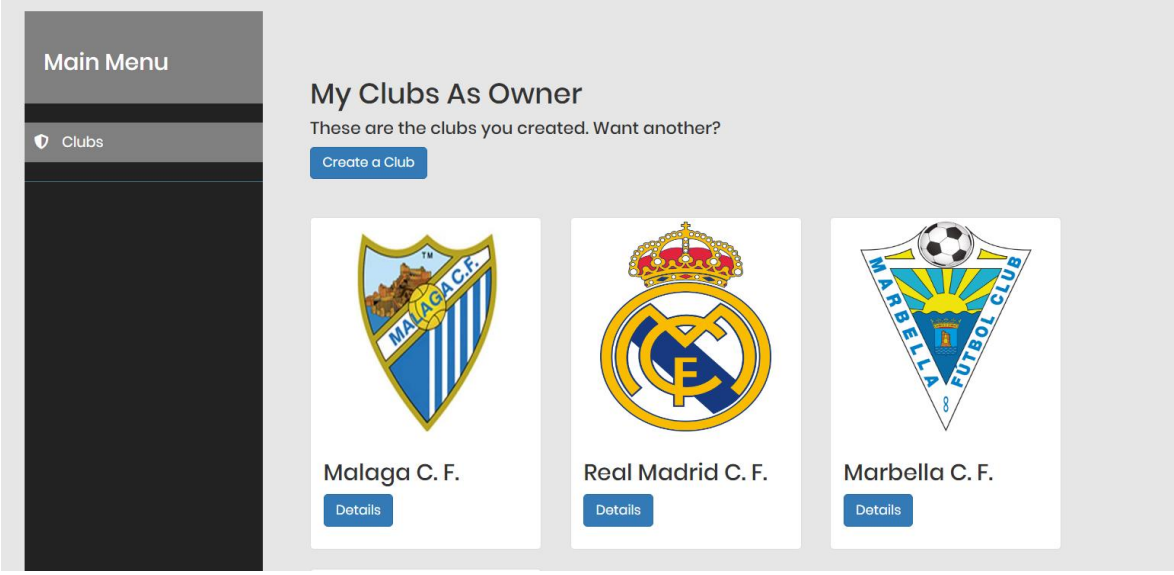


*Figura 47. Mis Clubs Como Dueño*

Si deseamos crear un club de nuestra propiedad, puede hacerse desde el botón “Create a Club” donde podremos introducir el nombre y la imagen del club para crearlo.

## 4. Gestión de club

Cuando entramos finalmente a un club pulsando en el botón “Details”, llegaremos a la vista de club. En ella podremos realizar todas las acciones relativas a la gestión de club, a las cuales se puede acceder mediante la barra de navegación lateral.

### 4.1 Detalles de club

Esta vista posibilitará al usuario editar los atributos del club, esto es, cambiar el nombre o la imagen. También se nos permitirá borrarlo, si así lo deseamos, pero debemos tener en cuenta que perderemos toda la información relativa al club, es decir, equipos, temporadas, jugadores, eventos y estadísticas.

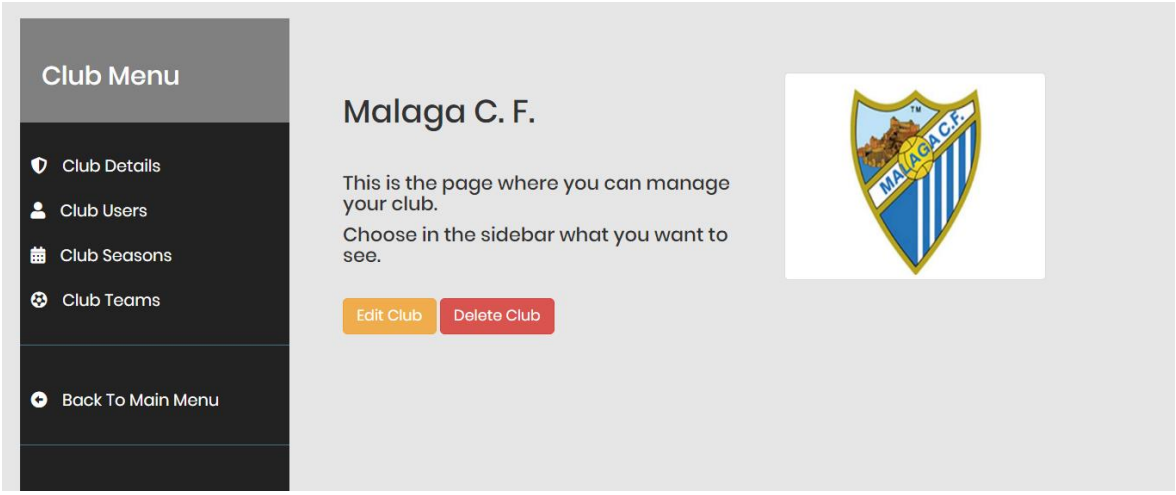


*Figura 48. Detalles de Club.*

## 4.2 Usuarios de club

Este apartado corresponde a la gestión de los usuarios de un club. La vista nos mostrará cuáles usuarios son dueños del club (administradores) y cuales son miembros (invitados).

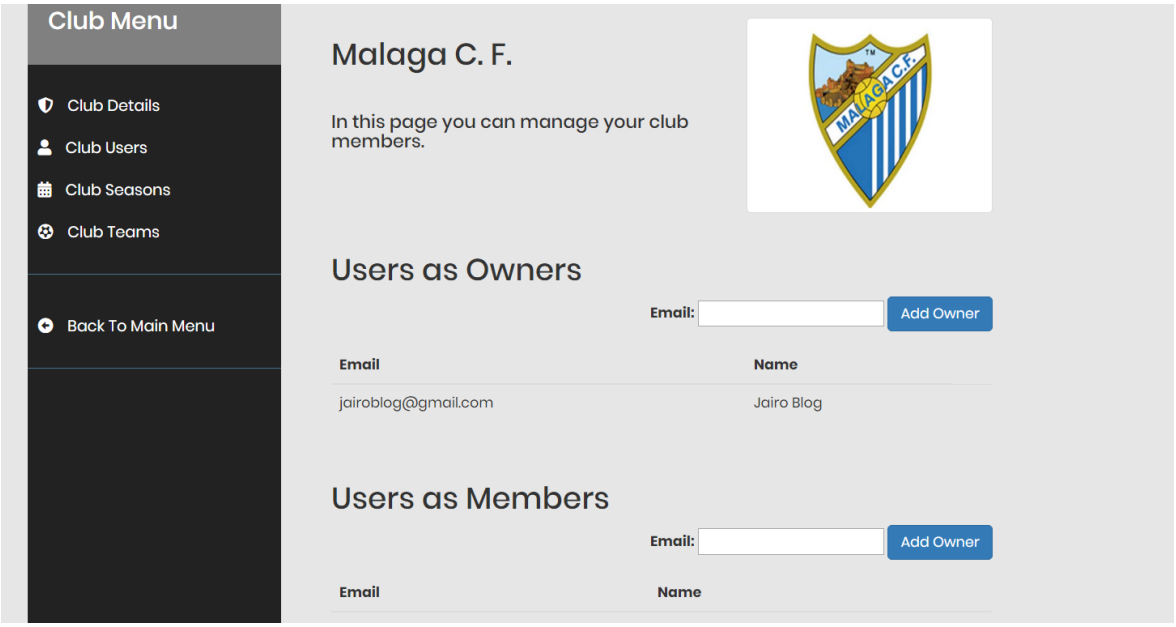


*Figura 49. Usuarios de Club.*

Para invitar a un usuario a colaborar en nuestro club, como dueño o como miembro, tan solo debemos escribir su correo electrónico en el apartado correspondiente y pulsar el botón “Add”.

Con ello, ese usuario al que invitamos podrá acceder a nuestro club cuando inicie sesión.

## 4.3 Temporadas de club

Este apartado nos permite definir temporadas para nuestro club. Las temporadas sirven para englobar a los equipos, los jugadores, los eventos y los partidos con el objeto de analizar el rendimiento de ellos en una temporada concreta.


*Figura 50. Temporadas de Club.*

Si queremos añadir una temporada al club tan solo debemos escribir el nombre y pulsar el botón “Add Season”.

En el listado de temporadas de club también podemos ver que cada una de ellas cuenta con un botón de borrado. Tenga en cuenta, que si elimina una temporada del sistema, también se eliminarán los jugadores, eventos y partidos sujetos a ella.

## 4.4 Equipos de club

Esta vista nos presentará los equipos pertenecientes al club.

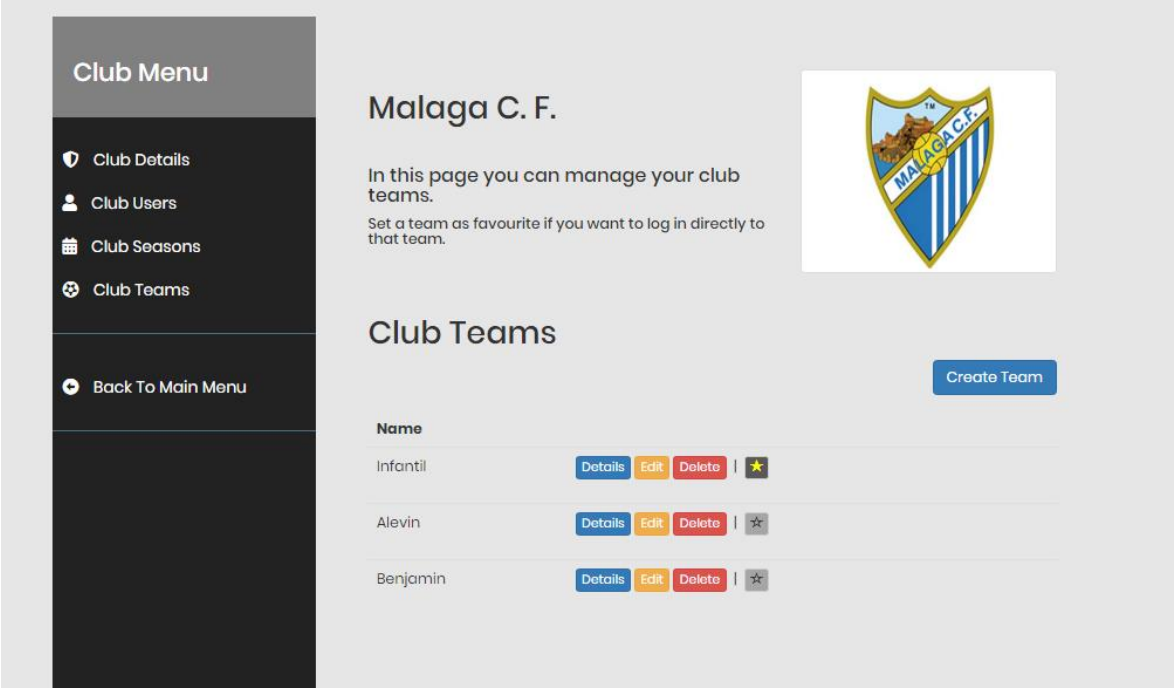


*Figura 51. Equipos de Club.*

Como podemos ver en la Figura 51, desde la vista podremos crear un equipo para el club. Cada equipo de la lista contará con botones para editarlo, borrarlo o hacerlo favorito.

Hay que tener en cuenta que si borramos un equipo, se eliminarán también sus jugadores, eventos y partidos asociados.

Si pulsamos el botón de la estrella, seleccionaremos ese equipo como favorito. Esto es útil para no tener que navegar hasta nuestro equipo habitual cada vez que iniciemos la aplicación, ya que, si marcamos un equipo como favorito, al iniciar sesión se nos llevará directamente a la vista de detalles del equipo.

Para entrar en un equipo y ver sus acciones asociadas debemos de pulsar en el botón “Details” en el equipo deseado.

# 5. Gestión de equipo

Si entramos en la vista de equipo, mediante el botón “Details”, se nos presentará una nueva barra de navegación lateral con las acciones relativas a a la gestión del equipo.

## 5.1 Jugadores de equipo

Aquí podremos acceder a los jugadores del equipo. Similar a las otras vistas, desde aquí podemos crear un nuevo jugador para el equipo pulsando el botón “Add Player”. Los jugadores se crearán como suplentes por defecto.

Cabe destacar también que los jugadores de un equipo deben estar sujetos a una temporada concreta.

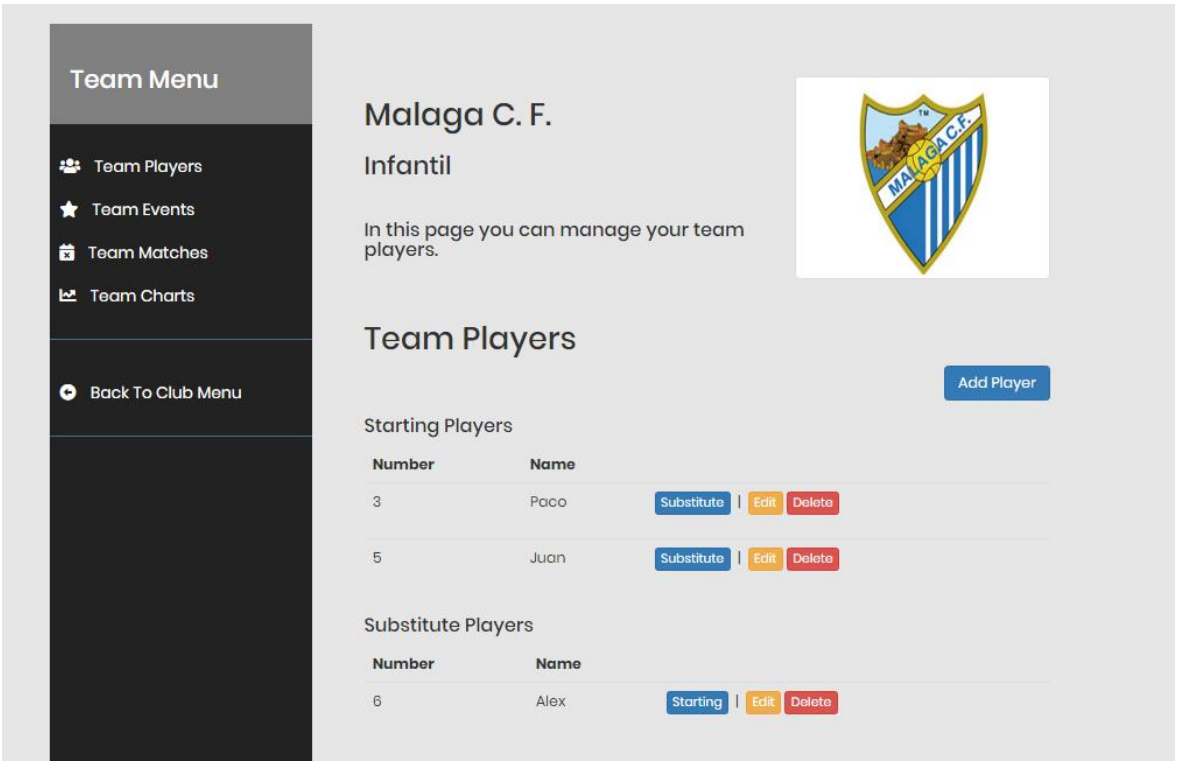


*Figura 52. Jugadores de Equipo.*

Aquí podremos ver dos listas de jugadores: los titulares y los suplentes. Mediante el botón azul, podremos pasarlos de uno a otro estado indistintamente.

También se nos proporcionan botones para editar y borrar al jugador. Hay que tener en cuenta que borrar a un jugador también eliminará las estadísticas de la temporada de ese jugador.

## 5.2 Eventos de equipo

En esta vista veremos los eventos del equipo. Mediante el botón “Add Event” podremos añadir nuevos eventos al equipo. Los eventos deben estar sujetos a una temporada concreta.

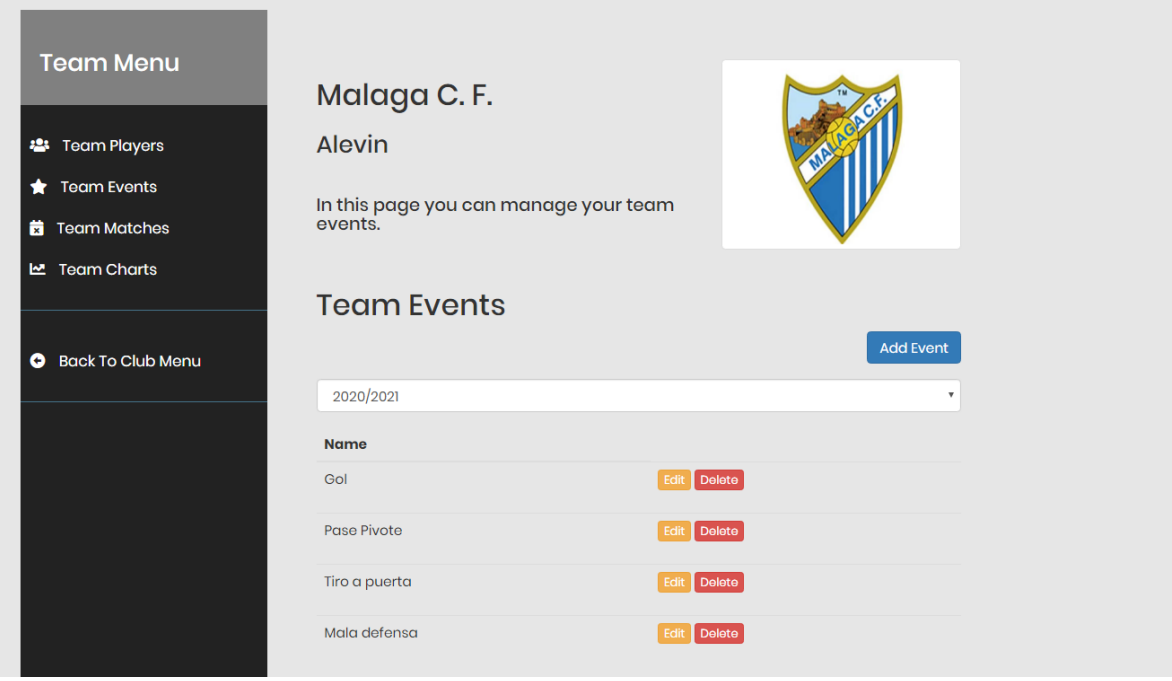


*Figura 53. Eventos de Equipo.*

Cada evento vendrá acompañado de botones para editarlo y borrarlo. Cabe destacar que borrar un evento también eliminará las estadísticas recogidas de ese evento en esa temporada.

### 5.3 Partidos de equipo

Esta es la vista relativa a la gestión de partidos de un equipo. Mediante el botón “Add Match” podremos crear un partido. Los partidos deben estar sujetos a una temporada concreta.

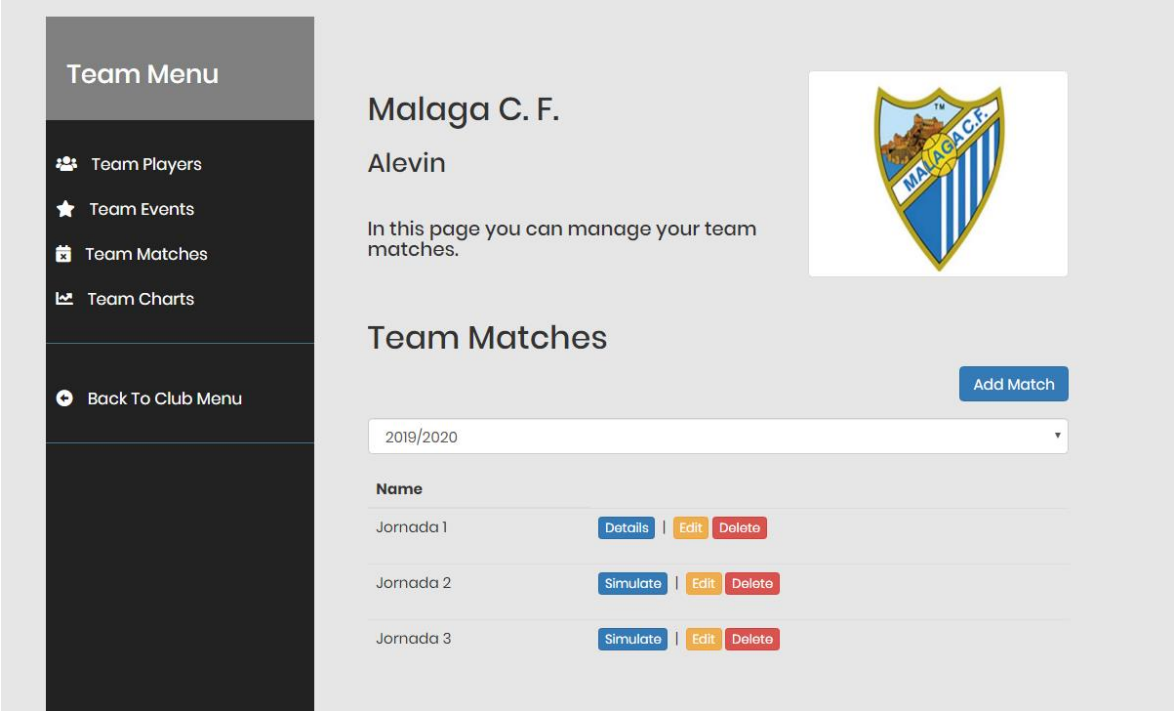


*Figura 54. Partidos de Equipo.*

Si un partido no ha sido jugado, se nos dará la opción de hacerlo con el botón “Simulate”. Si, por el contrario, ya se ha jugado, podremos ver los detalles del partido pulsando en el botón “Details”.

También tendremos la opción de editar y borrar un partido. Hay que tener en cuenta que borrar un partido también eliminará sus estadísticas de la temporada.

### 5.4 Estadísticas de equipo

Cuando pulsamos en el botón “Team Charts” de la barra de navegación lateral de equipo, podremos ver, por cada temporada, las estadísticas del equipo en forma de gráficos.

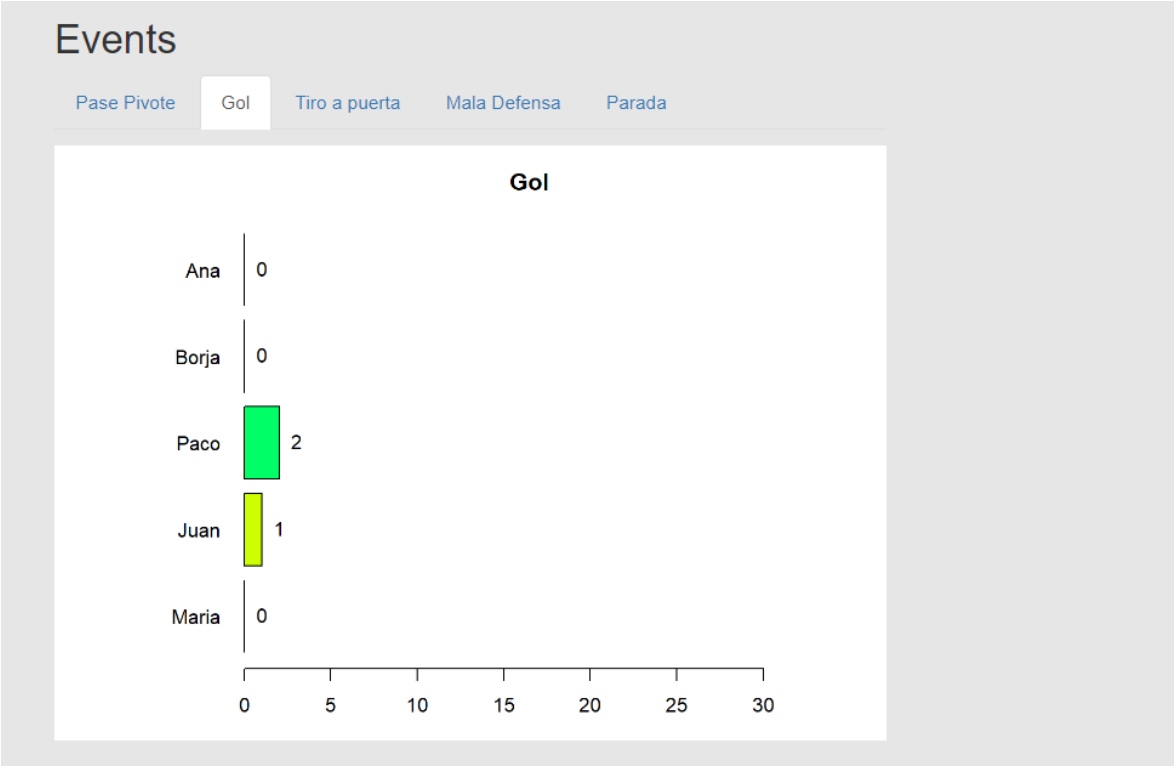


*Figura 55. Gráfico de Eventos.*

El primer conjunto de gráficas que podemos ver corresponde a las de evento, tal y como se muestra en la Figura 55. En ellas podremos ver, por cada evento, el número de veces que cada jugador lo ha ejecutado durante la temporada. Podremos navegar por las distintas pestañas para ver las gráficas de los distintos eventos.

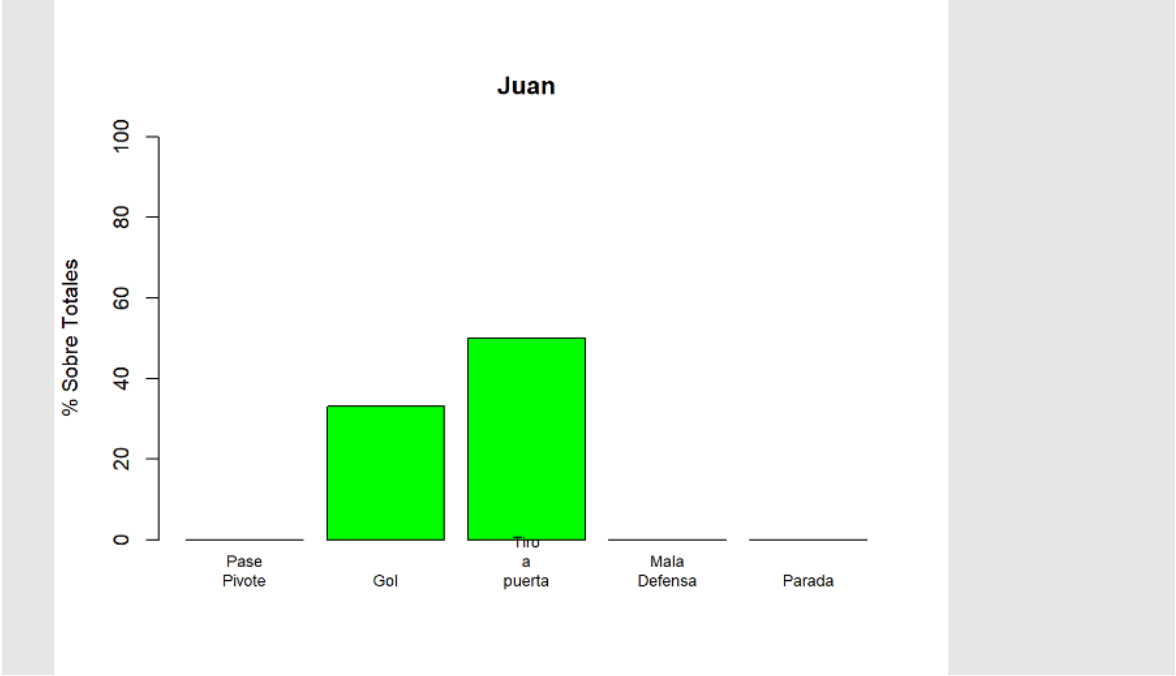


*Figura 56. Gráfico de Jugadores.*

En la Figura 56 podemos ver el segundo conjunto de gráficas que se muestran. Este conjunto corresponde a los eventos que cada jugador ha ejecutado sobre el porcentaje total del equipo. Podremos navegar hacia abajo en la página para ver las gráficas correspondientes a todos los jugadores.

## 6. Simulación de partidos

Si creamos un partido y pulsamos el botón “Simulate” se nos llevará a la vista de simulación de partido. A continuación se detallarán las distintas acciones que se pueden realizar durante un partido.

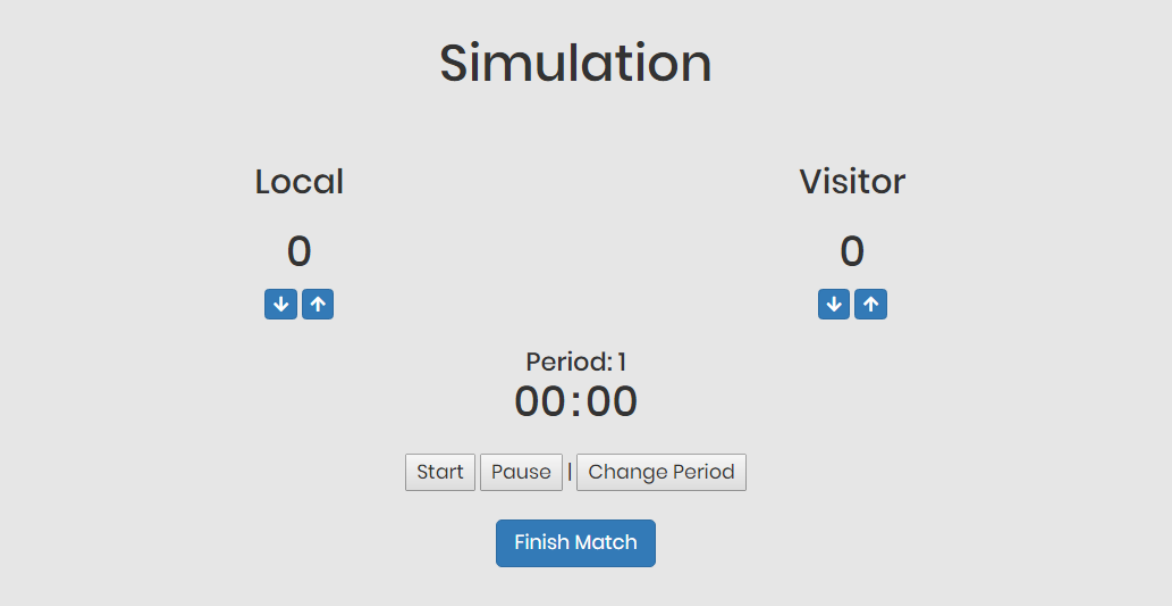


*Figura 57. Marcador de Partido.*

### 6.1 Manipular el marcador

En la vista de partido tendremos disponible el marcador local y el visitante, imprescindible para llevar la cuenta del resultado del partido.

Los goles o puntos se podrán añadir y quitar (tanto a un equipo como a otro) con los botones azules asociados. El botón con la flecha hacia arriba aumentará el marcador y el botón con la flecha hacia abajo lo disminuirá.

### 6.2 Iniciar y pausar cronómetro

En el centro de la parte superior podremos ver el cronómetro y justo debajo de él, botones para manipularlo. Para iniciarlo deberemos pulsar el botón “Start” y para pausarlo, el botón “Pause”.

### 6.3 Cambiar periodo

También se nos mostrará, justo encima del cronómetro, el periodo de partido, que se podrá también cambiar a nuestro gusto.

Si pulsamos el botón “Change Period” el periodo aumentará (no pudiendo volver atrás) y el cronómetro se reiniciará para dar comienzo a una nueva etapa del partido.

## 6.4 Finalizar partido

Justo debajo de todas estas acciones encontramos el botón de finalizar partido. Si lo pulsamos, se ejecutará el análisis estadístico del partido y la aplicación nos llevará a la vista de estadísticas de equipo.

## 6.5 Sustitución de jugadores

En cualquier momento del partido tendremos disponible la lista de jugadores titulares y suplentes. Podremos intercambiarlos indistintamente por medio de los botones azules asociados a cada jugador.


*Figura 58. Suplentes de Partido.*

El botón de la flecha hacia arriba de un jugador suplente, lo pondrá como titular y el botón de la flecha hacia abajo de un jugador titular lo pondrá como suplente.

## 6.6 Registro de ocurrencias de eventos

Esto representa la acción principal del partido. En una parte de la vista de partido tendremos a un lado los jugadores titulares y al otro, los posibles eventos que puedan ocurrir.


*Figura 59. Eventos de Partido.*

Si seleccionamos un jugador con el botón “Select Player” y un evento con el botón “Select Event” o viceversa, la ocurrencia del evento quedará registrada en el sistema.

## 6.7 Borrado de ocurrencias de eventos

En la vista de partido, siempre tendremos a mano el historial de ocurrencias. En él, podremos consultar todas las ocurrencias que hemos ido anotando durante el partido.

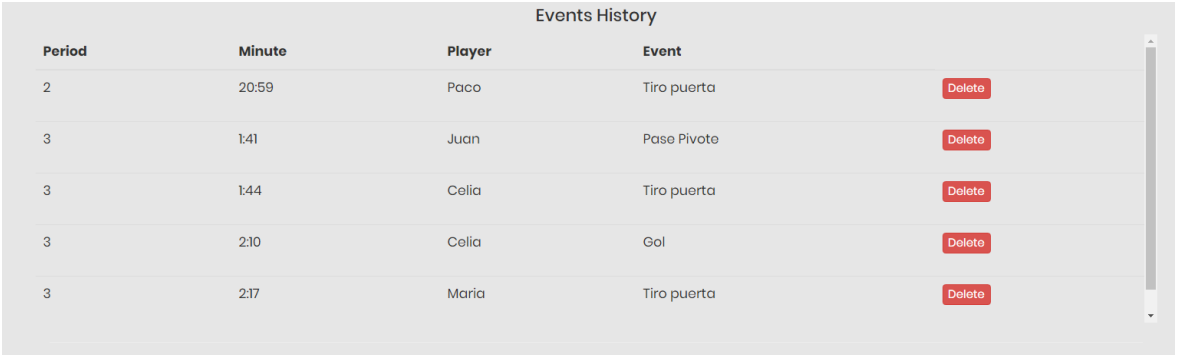


*Figura 60. Historial de Eventos.*

Si, por alguna razón, deseamos eliminar una ocurrencia, podremos hacerlo mediante el botón “Delete”. La ocurrencia quedará, entonces, fuera del registro y no se añadirá al análisis estadístico.